OIPE IAP22
MAY 13 2008
PATENT & TRADEMARK OFFICE

1F AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Barbara A. Christensen et al.

Serial No.: 09/822,675 Examiner: J. Hu

Filing Date: March 30, 2001 Group Art Unit: 2154

For: METHOD AND MECHANISM FOR THE DEVELOPMENT AND IMPLEMENTATION OF A WEB-BASED USER INTERFACE

Docket No.: 33012/312/101

TRANSMITTAL SHEET

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

CERTIFICATE UNDER 37 C.F.R. 1.8: I hereby certify that this correspondence and the documents described herein are being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to the: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 9th day of May, 2008

By _____
Carolyn I. Erickson

☐ A check in the amount of $_____ is enclosed.

☐ Small entity status of this application under 37 C.F.R. 1.9 and 1.27 has been established.

☒ Other: Supplemental Appeal Brief with Exhibits in Triplicate

☒ Please charge any deficiencies or credit any over payment in the enclosed fees to Deposit Account 14-0620.

By: _____
Wayne A. Sivertson
Reg. No. 25,645

NAWROCKI, ROONEY & SIVERTSON, P.A.
Suite 401, Broadway Place East
3433 Broadway Street N.E.
Minneapolis, Minnesota 55413
Telephone: (612) 331-1464
Facsimile: (612) 331-2239

OIPE IAP22
MAY 13 2008
PATENT & TRADEMARK OFFICE

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of	
Barbara A. Christensen et al.	Examiner J. HU
Serial No. 09/822,675	Group Art Unit 2154
Filing Date: 03/30/01	**SUPPLEMENTAL**
	APPEAL BRIEF
For: METHOD AND MECHANISM FOR THE DEVELOPMENT AND IMPLEMENTATION OF A WEB-BASED USER INTERFACE	

APPELLANT'S SUPPLEMENTAL APPEAL BRIEF
FILED UNDER 37 C.F.R. § 41.37

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

CERTIFICATE UNDER 37 C.F.R. 1.8: I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to the: Commissioner for Patents, Alexandria, VA, 22313-1450 on this

9th day of May, 20 08.

By _____
Carolyn I. Erickson

This supplemental appeal brief is being filed in triplicate within thirty days of mailing of the Notification of Non-Compliant Appeal Brief on April 9, 2008. Permission is hereby granted to charge or credit deposit account number 14-0620 for any errors in

fee calculation. Appellants request this Supplemental Appeal Brief be made of record and fully considered.

REAL PARTY IN INTEREST

The Real Party in interest is:

Unisys Corporation

Township Line and Union Meeting Roads

Blue Bell, Pennsylvania 19424

being the assignee of the entire right, title, and interest by all inventors, by way of assignment documents filed at Reel 011673, frame 0302, in the United States Patent and Trademark Office.

RELATED APPEALS AND INTERFERENCES

There are no known pending Appeals and/or Interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal. Therefore, there are no decisions to be placed in the attached Related Proceedings Appendix.

TABLE OF CONTENTS

STATUS OF CLAIMS

The subject patent application was filed on March 30, 2001 containing claims 1-20. Claims 1-2, 4-8, 10-13, 15-16, and 18-19 were amended by way of amendments filed December 19, 2004, August 3, 2005, September 6, 2005, March 24, 2006, August 14, 2006, November 13, 2006, and July 9, 2007. Claims 5, 10, 15, and 19-20 have been found to contain allowable subject matter.

Applicants herein appeal the final rejection of claims 1-4, 6-9, 11-14, and 16-18, which are in the state following entry of the amendment filed July 9, 2007. The Claims Appendix, hereto attached as an exhibit, reflects the finally rejected claims 1-4, 6-9, 11-14, and 16-18, along with allowable claims 5, 10, 15, and 19-20, following entry of the amendment filed July 9, 2007. Applicants herein appeal from the final rejection of claims 1-4, 6-9, 11-14, and 16-18, being all finally rejected pending claims.

STATUS OF THE AMENDMENTS

Applicant has filed responses to official actions on December 19, 2004, August 3, 2005, September 6, 2005, March 24, 2006, August 14, 2006, November 13, 2006, and July 9, 2007. The present appeal is from the final rejection of the Examiner mailed September 28, 2007.

AS a result, claims 1-4, 6-9, 11-14, and 16-18, recorded herewith as Claims Appendix, along with allowable claims 5, 10, 15, and 19-20, are in the form of July 9, 2007, following submission of the last presented amendment. This amendment is deemed to have been entered as a matter of right.

SUMMARY OF CLAIMED SUBJECT MATTER [1]

The present invention generally relates to data base management systems and more particularly relates to enhancements for dynamically building a customized user interface with the data base management system via the Internet[2].

In accordance with the present invention, a customized user interface is built for a given application which communicates with the application through the proprietary database management system. Unlike previous approaches, the customized user interface stores inputs to the application within a predesignated area of the proprietary database management system. The application accesses these inputs as data requests. Similarly, the application provides the results to the proprietary database management system from which they are retrieved by the customized user interface for transmission to the user terminal. Thus, the customized user interface and corresponding application communicate with each other only indirectly through the proprietary database management system.[3]

---

[1] The references to the specification and drawings provided herein are only exemplary and are not deemed to be limiting. The purpose of the references is to enable the Board to more quickly determine where the claimed subject matter is described within the present application.

[2]See Specification at page 5, lines 2-5.

[3]See Specification at page 14, line 20, through page 15, line 8.

This approach enables a first software engineer to focus on the logic of the application component without the distractions of designing and/or interfacing to a customized user interface. All inputs to and outputs from the application component are essentially accesses of the proprietary database management system.[4]

Similarly, a user interface designer is permitted to focus exclusively on the customized user interface. Likewise, all server side inputs to and outputs from the customized user interface are simply proprietary database management accesses. As a result, all interactions of the customized user interface and the corresponding application are predefined by the access rules of the proprietary database management system.[5]

Fig. 3 is a pictorial diagram of hardware suite 44 of the preferred embodiment of the present invention[6]. Fig. 26A is a schematic diagram showing a system with direct communication between a customized user interface and corresponding application[7].

Fig. 26B is a similar schematic diagram showing no direct interface between customized user interface 842 and application 846. It is to be noted that there is no direct interface between these two elements with specialized 844 not being present (see also

---

[4]See Specification at page 15, lines 9-13.
[5]See Specification at page 15, lines 14-19.
[6]See Specification at page 23, lines 3-4.
[7]See Specification at page 53, line 2, through page 54, line 1.

Fig. 26A). As a result, customized user interface 842 and corresponding application 846 are developed independently, without undue focus on the other.[8]

Instead, all communication between customized user interface 842 and corresponding application 846 is via proprietary database management system 852. User data is transferred via path 854 to database management system 852 for temporary storage until accessed by corresponding application 846 via path 848. Similarly, data to be sent to the user is temporarily stored within database management system 852 via path 850 and retrieved by customized user interface 842 via path 856.[9]

Claim 16 is the only pending claim having "means-plus-function" limitations. It has four such limitations which are correlated to Applicants' disclosure as follows:

a) "permitting means"[10];

b) "providing means"[11];

c) "offering means"[12]; and

d) "processing means"[13].

Claims 17-20 which depend from independent claim 16 present no additional "means-plus-function" limitations.

---

[8]See Specification at page 54, lines 14-19.
[9]See Specification at page 54, line 20, through page 55, line 5.
[10]See Specification at page 23, lines 4-8, and Fig. 3, Client 46.
[11]See Specification at page 54, lines 1-5, and Figs. 26A and 26B, element 842.
[12]See Specification at page 24, lines 3-6, and Fig. 3, element 54.
[13]See Specification at page 54, line 20, through page 55, line 5 , and Fig. 26B, element 846.

In accordance with the Notification of Non-Compliant Appeal Brief mailed March 27, 2008, Applicants herewith endeavor to map claims 1, 6, and 11 to "the specification by page and line number or paragraph number and/or drawings, if any".

Claim 1:

---preamble --- see Figs. 3, 26A, and 26B, and specification at pages 23-24 and 54; and

--- element a -- see Fig. 26B, element 846, and specification at page 54, line 20, through page 55, line 5.

Claim 6:

--- element a -- see Fig. 3, element 46, and specification at page 23, lines 4-8;

--- element b -- see Figs. 26A and 26B, element 842, and specification at page 54, lines 1-5;

--- element c -- see Fig. 3, element 54, and specification at page 24, lines 3-6; and

--- element d -- see Fig. 26B, element 846, and specification at page 54, line 20, through page 55, line 5.

Claim 11:

--- element a -- see Fig. 3, element 46, and specification at page 23, lines 4-8;

--- element b -- see Figs. 26A and 26B, element 842, and specification at page 54, lines 1-5;

--- element c -- see Fig. 3, element 54, and specification at page 24, lines 3-6; and

--- element d -- see Fig. 26B, element 846, and specification at page 54, line 20, through page 55, line 5.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Are claims 1-4, 6-9, 11-14, and 16-18, unpatentable under 35 U.S.C. 103(a) as being obvious over U.S. Patent Application Publication No. 2002/0026507, published in the name of Sears et al (hereinafter referred to as "Sears") in view of U.S. Patent No. 6,643,690, issued to Duursma et al. (hereinafter referred to as "Duursma")?

ARGUMENT

I. Claims 1-4, 6-9, 11-14, and 16-18, are not unpatentable under 35 U.S.C. 103(a) as being obvious over Sears in view of Duursma.

Claims 1-4, 6-9, 11-14, and 16-19 have been rejected under newly applied 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Publication No. 2002/0026507, issued in the name of Sears et al (hereinafter referred to as "Sears") in view of U.S. Patent No. 6,643,690, issued to Duursma et al (hereinafter referred to as "Duursma"). This ground of rejection is respectfully traversed for the following reasons.

Because it is dispositive of the pending prior art rejections Applicants filed a first declaration under 37 C.F.R. 1.131 (copy enclosed herewith in the Evidence Appendix), along with an Amendment After Final on August 14, 2006 in an attempt to swear behind Sears, being the primary reference. This declaration is based upon the fact that Applicants' invention as disclosed and claimed is incorporated into a commercial product (i.e., "Cool ICE, Revision 2.1") of Unisys Corporation, assignee of the subject application. Actual reduction to practice and thus completion of the invention is established upon commercialization of the product containing the invention on April 4, 2000.

On August 29, 2006, the Examiner mailed an Advisory Action stating that the Amendment After Final was entered without specific reference to the first declaration, without establishing allowability of the claimed invention. Thus, Applicants assumed that the first declaration had been entered into the record and filed a first Notice of Appeal, along with a request for pre-appeal brief conference. On October 23, 2006, SPE John Follansbee mailed a Notice of Panel Decision from Pre-Appeal Brief Review indicating that the Examiner had changed his mind about entry of the first declaration.

In response thereto, Applicants filed a Request for Continued Examination on November 13, 2006 in an attempt to have the first declaration entered into the record. On January 9, 2007, the Examiner mailed a non-final official action stating that the first declaration was insufficient because:

> There is **not enough evidence** to clearly prove the relationships between "Cool ICE Revision 2.1" and the claims. (emphasis added)

Thus, the Examiner continued his rejections in view of Sears, apparently deeming that the "Cool ICE, Revision 2.1" product was on sale in April 2000 (i.e., a matter of public record when a publicly traded Fortune 500 Company, such as Unisys Corporation, releases a major product having a material impact upon its financial situation in addition to the sworn testimony of Applicants/Declarants). However, the Examiner alleges "**not enough evidence**" to show that

the claimed invention is incorporated into the Cool ICE Revision 2.1 product.

Enclosed herewith is the Supplemental Declaration of the Inventors of Record under 37 C.F.R. 1.131 establishing of record that the subject invention was made before the effective filing date (i.e., August 4, 2000) of Sears). As indicated by the attached declaration and as a matter of public record, Applicants' invention was completely conceived, reduced to practice, and commercially on-sale as of April 4, 2000. This Supplemental Declaration has been submitted because the Examiner has rejected the previously submitted declaration under 37 C.F.R. 1.131. Thus, the enclosed Supplemental Declaration is directed to showing that Cool ICE Revision 2.1 was on-sale in April 2000 and that the subject commercial product contained the invention of pending claims 1-20. Specifically, the Examiner's attention was directed to Section 10 of the attached copy of Unisys Corporation Document 7850 2473-001 (bearing an April 2000 copyright date), which discusses operation of the Data Wizard. Also pertinent are Sections 2, 4, 5, 9, 11, and 12, of the document which discuss the overall architecture, component definitions, objects, and the unique and novel user interface as claimed.

It is emphasized that this Supplemental Declaration establishes that the invention of claims 1-20 was embodied in a commercial, on-sale product of Unisys Corporation, assignee of the

subject application. Therefore, the invention of pending claims 1-20 were of necessity completely conceived and reduced to practice at least as early as April 2000.

In his final office action mailed September 28, 2007, the Examiner rejected Applicants' Supplemental stating in part:

> However, in the affidavits (sic) the applicant recited "Cool ICE Revision 2.1 was completed and first placed on commercial sale on 04/04/00" which is a mere allegation.

This statement is clearly erroneous and incorrect as a matter of law.

It is clearly erroneous, because Applicants' statements are not "mere allegations" but sworn statements. As a result, such sworn statements are punishable under the criminal laws of the United States of America, if incorrect.

Furthermore, such sworn statements are "evidence" as a matter of law. Therefore, such "evidence" shifts the burden of proof to the Examiner to establish the inaccuracy of the assertion. Of course, the Examiner could not do so, because the release of the product is a matter of public record. A publicly traded company, such as Unisys Corporation, must release anticipated and actual release dates for products which have a "material" impact upon the financial condition of the company. Furthermore, the commercial release must be made publicly for marketing purposes.

In rejecting Applicants' Supplemental Declaration, the Examiner further states:

> There is not enough evidence to clearly prove the relationships between "Cool ICE Revision 2.1" and the claims. The applicant has just only pointed out that "The document of Exhibit A in general and sedtions (sic) 2, 4, 5, and 9-12, in particular, establish that the commercial product describe therein (i.e., Cool ICE Revision 2.1), which was on-sale in April 2000, contains all of the elements and the combination thereof constituting the invention of claims 1-20 of the subject application." The applicant fails to concisely/specifically explained (sic)/describe/mapped (sic) the elements of each of sections 2, 4, 5, and 9-12 corresponding to the limitations of claims.

Apparently, the Examiner fails to understand the nature of the Cool ICE product and Applicants' claimed invention.

The Cool ICE product is a very substantial system costing millions of dollars having millions of lines of script and many separate hardware elements distributed over a relatively large geographical area. The documentation commercially available for the Cool ICE system is indeed substantial, even though dwarfed by the confidential internal documentation within Unisys Corporation, commercial supplier of the system. However, the subject application is the only single document dedicated to enablingly disclose all of the elements of Applicants' claimed invention.

Therefore, to spare all of the individuals interested in the prosecution history of the subject application the difficulties associated with managing a literal blizzard of documents, Applicants have elected to support their Supplemental Declaration using the Technical Overview submitted as Exhibit A to their Supplemental Declaration. Whereas this document contains in excess

of 200 pages of information, it is admitted that the document does not enablingly disclose all elements of the claimed invention. However, the Examiner cites no authority to support his finding of the requirement to "concisely/specifically explained (sic)/describe/mapped (sic) the elements of each of sections 2, 4, 5, and 9-12 corresponding to the limitations of claims".

The Supplemental Declaration of Applicants is the sworn statement and therefore evidence that Applicants' invention as disclosed and claimed is incorporated into the Cool ICE commercial product of Unisys Corporation, and that commercial product was on-sale at least as early as April 2000. Exhibit A of the Supplemental Declaration supports this sworn statement that Cool ICE, Revision 2.1 was on-sale at least as early as April 2000, and Applicants' claimed invention is included in that product. As a result, one is compelled to find that Applicants' claimed invention was actually reduced to practice at least as early as April 2000 and thus Sears is not prior art to Applicants' invention as disclosed and claimed.

Even though Applicants' first declaration and Supplemental Declaration are sufficient to conclude that pending claims 1-20 completely distinguish over the prior art rejections of the Examiner, these rejections are further traversed in view of the Examiner's failure to present a *prima facie* case of obviousness as

specified by MPEP 2143 even if Sears were not removed as a reference against the pending claims.

To make a *prima facie* case of obviousness, MPEP 2143 requires the Examiner to provide evidence and argument showing: 1)motivation to make the alleged combination; 2)reasonable likelihood of success of the alleged combination; and 3)all claimed elements within the alleged combination. The Examiner has failed to make any of these three required showings. Therefore, because the Examiner has not made a *prima facie* case of obviousness, Applicants need not and indeed cannot offer appropriate evidence and argument in rebuttal.

The first required showing is that of motivation. In an apparent attempt to comply with this requirement, the Examiner states:

> Sear (sic) does not specifically teach user interface module located within the database of the data base management system.

Without any explanation, the Examiner bases his finding of motivation by directly contradicting a prior finding from his previous official action mailed June 13, 2006, which states in part at paragraph 4:

> As per claims 1-3, Sears teaches, ... a user interface module ... located within said data base of said data base management system .... [para. 98-99].

Thus, the Examiner's alleged assertion of motivation is at least based upon clearly erroneous and contradictory findings[14].

In further support of his assertion of motivation, the Examiner states:

> It would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify aaa's (sic) system with Duursma's service module because doing so would improve the service quality by allowing the database management system easily (sic) transit data to other database management systems [Duursma, col. 3, lines 33-40].

This statement is inadequate. There is no showing that Sears does not "transmit data to other database management systems". In fact, Sears explicitly discloses that it has this capability. Therefore, the alleged combination would not be motivated as superfluous.

Furthermore, Duursma, column 3, lines 33-40, says nothing of "easily transmit data". Thus, the Examiner's alleged reason to make the alleged combination of Sears and Duursma is not supported by the prior art of record. As a result, motivation has not been shown in accordance with controlling law.

The second showing required by MPEP 2143 is that of reasonable likelihood of success. Even though this is a required showing, the Examiner simply ignores his obligation in this regard. In view of the readily apparent incompatibilities between Sears and Duursma,

---

[14]To the extent the earlier finding was correct, the Examiner has based his showing of motivation on a clearly erroneous finding of fact. To the extent the earlier finding was incorrect, the Examiner has explicitly admitted that his previous final rejection was erroneous as a matter of law and the resulting RCE filed by Applicants was a needless waste of resources.

the Examiner could not show reasonable likelihood of success, even if he had tried.

The final showing is that of all claimed elements within the alleged combination. The essence of the present invention as disclosed and claimed requires a user terminal coupled to a data base management system via a publically accessible digital data communication network having a customized user interface module which is itself coupled to the user terminal via the publically accessible digital data communication network. The claimed customized user interface module is **not** in the claimed user terminal. The importance of this architecture is explained in detail in Applicants' specification providing various types of customized user interfaces associated with the various applications.

This feature was carefully pointed out to the Examiner in Applicants' previous submissions. Nevertheless, the Examiner cites Duursma as yet additional prior art wherein the "client" hosts its own "user interface module" providing the user interface. Nevertheless, the Examiner makes extensive citations from Duursma concerning the "service module", located within the server, which is not the "user interface module" of Duursma, much less the claimed "user interface module".

I.A. Claim 1 is not unpatentable over Sears in view of Duursma.

With regard to claim 1, for example, the Examiner cites Fig. 2 of Sears to show the claimed "a user terminal for accessing a selected one of a plurality of applications coupled to a data base management system having a data base". Yet, Fig. 2 shows no "data base management system having a data base" as claimed.

Claim 1 is also limited by "a user interface module coupled to said user terminal via said publicly accessible digital data communication network and located within said data base of said data base management system which communicates with said selected one of said plurality of applications only via said data base management system". In making his rejection, the Examiner cites paragraphs 0098-0099 of Sears, which discusses the contents of Memory 14 shown in Fig. 3. Memory 14 of Fig. 3 is shown as the internal memory of computer 11 of Fig. 1. However, as cited above, computer 11 is really individual user 78. Therefore, in accordance with the disclosure of Sears as cited by the Examiner, all of the claimed components are co-located within the computer 11 which is really individual user 78. Furthermore, Duursma does not disclose any "user interface module" located within the data base management system, as claimed.

In paragraph 6, at page 4, of his final rejection, the Examiner admits that Sears does not have the claimed "user

interface module" located within the database of a data base management system as claimed. Therefore, he states:

> ....Duursma on the other hand teaches the **service module**..... (emphasis added)

This finding is legally irrelevant, because it does not address Applicants' claimed invention. The claimed element is a "user interface module", not the "service module" disclosed by Duursma. The "user interface 56" of Duursma is clearly located within client 10 as shown in Fig. 4, rather than in the data base of a data base management system as claimed by Applicants.

The rejection of claim 1, and all claims depending therefrom, should be reversed for failure of the Examiner to make any of the three showings required by MPEP 2143.

I.B. Claim 2 is not unpatentable over Sears in view of Duursma.

Claim 2 depends from claim 1 and further limits the claimed "publicly accessible digital data communication network". Because the alleged combination does not meet the limitations of claim 1, it cannot meet the further limitations of claim 2. The rejection of claim 2 should be reversed.

I.C. Claim 3 is not unpatentable over Sears in view of Duursma.

Claim 3 depends from claim 2 and further limits the software architecture of the "User Terminal". Sears says nothing of software architecture of any computer and certainly shows nothing of software architecture in Fig. 2. The rejection of claim 3 should be reversed as based upon clearly erroneous findings of fact.

I.D. Claim 4 is not unpatentable over Sears in view of Duursma.

Claim 4 depends from claim 3 and further requires that the "application" and the "user interface module" are located within a single server. This limitation could only be met by Sears if all claimed components are located within computer 11.

Perhaps more interesting is that the Examiner, having admitted in his rejection of claim 1 that Sears does not meet the claimed "user interface module" element, finds that Sears does meet these requirements in rejecting claim 4. As a result, the rejection of claim 4 should be reversed.

I.E. Claim 6 is not unpatentable over Sears in view of Duursma.

In rejecting claim 6, the Examiner clearly erroneously states:

> As per claims 6-9, since (sic) they are apparatus claims of claims 1-4, they are rejected for the same basis as claims 1-4 above.

Claims 1-4 and 6-9 are not of the same scope. For example, claim 6 is limited by "an application responsively coupled to said user interface module via said data base management system". This limitation is simply not found in claims 1-4. Therefore, in addition to the clearly erroneous findings of fact and errors of law with regard to the rejection of claims 1-4, the rejection of claims 6, and all claims depending therefrom, should be reversed for being incomplete as a matter of controlling law.

I.F. Claim 7 is not unpatentable over Sears in view of Duursma.

Claim 7 depends from claim 6 and further limits the coupling network. The Examiner has failed to examine claim 6, therefore, he could not possibly address the further limitations of claim 7. Therefore, in addition to the clearly erroneous findings of fact with regard to the rejection of claims 1-4, the rejection of claim 7 should be reversed for being incomplete as a matter of controlling law.

I.G. Claim 8 is not unpatentable over Sears in view of Duursma.

Claim 8 depends from claim 7 and is further limited by elements not found in claims 1-4. Nevertheless, in rejecting claim 8, the Examiner clearly erroneously states:

> As per claims 6-9, since (sic) they are apparatus claims of claims 1-4, they are rejected for the same basis as claims 1-4 above.

Therefore, in addition to the clearly erroneous findings of fact with regard to the rejection of claims 1-4, the rejection of claim 8 should be reversed for failure of the Examiner to actually examine the claim in accordance with controlling law.

I.H. Claim 9 is not unpatentable over Sears in view of Duursma.

In rejecting claim 9, the Examiner clearly erroneously states:

> As per claims 6-9, since (sic) they are apparatus claims of claims 1-4, they are rejected for the same basis as claims 1-4 above.

Claims 1-4 and 6-9 are not of the same scope. For example, claim 6 is limited by "an application responsively coupled to said user interface module via said data base management system". This limitation is simply not found in claims 1-4. Therefore, in addition to the clearly erroneous findings of fact with regard to the rejection of claims 1-4, the rejection of claim 9 should be reversed as incomplete as a matter of controlling law.

I.I. Claim 11 is not unpatentable over Sears in view of Duursma.

In rejecting claim 11, the Examiner clearly erroneously states:

> As per claims 11-14, since (sic) they are method claims of claims 1-4, they are rejected for (sic) the same basis as claims 1-4 above.

Claim 11, for example, is a method claim having four steps. The alleged combination has none of these four steps. Furthermore, the Examiner has not alleged that the alleged combination does have these four steps, because he continues to refuse to even examine these claims in opposition to controlling law.

For example, claim 11 requires "transferring", "receiving", "storing", and "retrieving". The alleged combination does not have any of these steps as claimed. In addition to the clearly erroneous findings of fact and clear errors of law identified above, the rejection of claim 11, and all claims depending therefrom, should be reversed for failure to be examined in accordance with controlling law.

I.J. Claim 12 is not unpatentable over Sears in view of Duursma.

In rejecting claim 12, the Examiner clearly erroneously states:

> As per claims 11-14, since (sic) they are method claims of claims 1-4, they are rejected for (sic) the same basis as claims 1-4 above.

Claim 12 is a method claim depending from claim 11 having four additional limiting steps. The alleged combination has none of these four steps. Furthermore, the Examiner has not alleged that

the alleged combination does have these four steps, because he continues to refuse to even examine these claims in opposition to controlling law.

For example, claim 12 requires "processing", "storing", "accessing", and "transferring" steps. The alleged combination does not have any of these steps as claimed. In addition to the clearly erroneous findings of fact and clear errors of law identified above, the rejection of claim 12 should be reversed as being not having been examined.

I.K. Claim 13 is not unpatentable over Sears in view of Duursma.

In rejecting claim 13, the Examiner clearly erroneously states:

> As per claims 11-14, since (sic) they are method claims of claims 1-4, they are rejected for (sic) the same basis as claims 1-4 above.

Claim 11, for example, is a method claim having four steps. The alleged combination has none of these four steps. Furthermore, the Examiner has not alleged that the alleged combination does have these four steps, because he continues to refuse to even examine these claims in opposition to controlling law.

For example, claim 11 requires "transferring", "receiving", "storing", and "retrieving". The alleged combination does not have any of these steps as claimed. Claim 13 contains yet additional

limitations. In addition to the clearly erroneous findings of fact identified above, the rejection of claim 13 should be reversed as being incomplete as a matter of law.

I.L. Claim 14 is not unpatentable over Sears in view of Duursma.

In rejecting claim 14, the Examiner clearly erroneously states:

> As per claims 11-14, since (sic) they are method claims of claims 1-4, they are rejected for (sic) the same basis as claims 1-4 above.

Claim 11, for example, is a method claim having four steps. The alleged combination has none of these four steps. Furthermore, the Examiner has not alleged that the alleged combination does have these four steps, because he continues to refuse to even examine these claims in opposition to controlling law.

For example, claim 11 requires "transferring", "receiving", "storing", and "retrieving". The alleged combination does not have any of these steps as claimed. In addition to the clearly erroneous findings of fact identified above, the rejection of claim 14 should be reversed as being incomplete as a matter of law.

I.M. Claim 16 is not unpatentable over Sears in view of Duursma.

In rejecting claim 16, the Examiner again seeks to avoid his duty and fails to apply controlling law. Failing to acknowledge

the difference in statutory basis and the examination procedures mandated by MPEP 2181 et seq., the Examiner states:

> As per claims 16-19, since (sic) they are means plus function claims of claims 1-4, they are rejected for (sic) the same basis as claims `1-4 above.

In addition to the clearly erroneous findings of fact explained above, this ground of rejection is clearly incorrect as a matter of law. The rejection of claim 16, and all claims depending therefrom, should be reversed for failure of the Examiner to comply with controlling law.

I.N. Claim 17 is not unpatentable over Sears in view of Duursma.

In rejecting claim 17, the Examiner again fails to apply controlling law. Failing to acknowledge the difference in statutory basis and the examination procedures mandated by MPEP 2181 et seq., the Examiner states:

> As per claims 16-19, since (sic) they are means plus function claims of claims 1-4, they are rejected for (sic) the same basis as claims `1-4 above.

In addition to the clearly erroneous findings of fact explained above, this ground of rejection is clearly incorrect as a matter of law. The rejection of claim 17 should be reversed for failure of the Examiner to apply controlling law.

I.O. Claim 18 is not unpatentable over Sears in view of Duursma.

In rejecting claim 18, the Examiner again fails to apply controlling law. Failing to acknowledge the difference in statutory basis and the examination procedures mandated by MPEP 2181 et seq., the Examiner states:

> AS per claims 16-19, since (sic) they are means plus function claims of claims 1-4, they are rejected for (sic) the same basis as claims `1-4 above.

In addition to the clearly erroneous findings of fact explained above, this ground of rejection is clearly incorrect as a matter of law. The rejection of claim 18 should be reversed for failure to be examined in accordance with controlling law.

CONCLUSION

Having thus reviewed the final rejections of claims 1-4, 6-9, 11-14, and 16-18, being all finally rejected pending claims, it seems abundantly clear that the limitations of these claims are not unpatentable in view of the prior art of record. Thus, the rejection of these claims should be reversed as being based upon clearly erroneous fact findings and errors of law.

Respectfully submitted

Barbara A. Christensen et al.

By their attorney,

Date May 9 2008

Wayne A. Sivertson
Reg. No. 25,645
Suite 401,Broadway Place East
3433 Broadway Street N.E.
Minneapolis, Minnesota 55413
(612) 331-1464

CLAIMS APPENDIX

1. In a data processing system having a user terminal for accessing a selected one of a plurality of applications coupled to a data base management system having a data base responsively coupled to said user terminal via a publicly accessible digital data communication network, the improvement comprising:

a. a user interface module coupled to said user terminal via said publicly accessible digital data communication network and located within said data base of said data base management system which communicates with said selected one of said plurality of applications only via said data base management system.

2. The improvement according to claim 1 wherein said publicly accessible digital data communication network further comprises the Internet.

3. The improvement according to claim 2 wherein said user terminal further comprises an industry compatible personal computer having a commercially available browser.

4. The improvement according to claim 3 wherein said selected one of said plurality of applications, said data base management system, and said user interface module are resident within a single server.

5. The improvement according to claim 4 wherein said data base management system is Classic MAPPER data base management system.

6. An apparatus comprising:

a. a user terminal;

b. a user interface module responsively coupled to said user terminal via a publicly accessible digital data communication network;

c. a data base management system responsively coupled to said user interface module and containing said user interface module; and

d. a plurality of applications responsively coupled to said user interface module via said data base management system.

7. The apparatus of claim 6 wherein said publicly accessible digital data communication network further comprises the Internet.

8. The apparatus of claim 7 wherein said user terminal communicates with said plurality of applications via said Internet, said user interface module, and said data base management system.

9. The apparatus of claim 8 wherein said user terminal further comprises an industry compatible personal computer containing a web browser.

10. The apparatus of claim 9 wherein said data base management system further comprises the Classic MAPPER data base management system.

11. A method of communicating between a user terminal coupled via a publicly accessible digital data network to an application located on a remote server having a data base management system with a data base comprising:

a. transferring a service request from said user terminal to said remote server via said publicly accessible digital data network;

b. receiving said service request with a user interface module located within said remote server;

c. storing said service request within said data base management system; and

d. retrieving said service request from said data base management system by said application.

12. A method according to claim 11 further comprising:

e. processing said service request with said application to produce a response;

f. storing said response in said data base management system;

g. accessing said response from said data base management system by said user interface module; and

h. transferring said response to said user terminal from said user interface module to said user terminal via said publicly accessible digital data communication network.

13. A method according to claim 12 wherein said publicly accessible digital data communication network further comprises the world wide web.

14. A method according to claim 13 wherein said user terminal further comprises an industry compatible personal computer.

15. A method according to claim 14 wherein said remote data base management system further comprises Classic MAPPER data base management system.

16. An apparatus comprising:

a. permitting means for permitting a user to access a publicly accessible digital data communication network;

b. providing means responsively coupled to said permitting means via said publicly accessible digital data communication network for providing a user interface to said permitting means;

c. offering means responsively coupled to said providing means for offering data base management services; and

d. processing means responsively coupled to said offering means for processing data applications which communicates with said permitting means via said offering means.

17. An apparatus according to claim 16 wherein said permitting means and said processing means are coupled only via said offering means.

18. An apparatus according to claim 17 wherein said publicly accessible digital data communication network further comprises the Internet.

19. An apparatus according to claim 18 wherein said responding means further comprises Classic MAPPER data base management system.

20. An apparatus according to claim 19 wherein said permitting means further comprises an industry standard personal computer.

RELATED EVIDENCE APPENDIX

This Appendix contains a Declaration under 37 C.F.R. 1.131 filed August 14, 2004 and re-filed November 13, 2006, along with a Supplemental Declaration under 37 C.F.R. 1.131 and exhibits filed July 9, 2007.

RELATED PROCEEDINGS APPENDIX

There are no decisions or other papers deemed appropriate to be included in this Appendix.

OIPE IAP22
MAY 13 2008
PATENT & TRADEMARK OFFICE

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	
Barbara A. Christensen et al.	Examiner J. Hu
Serial No. 09/822,675	Group Art Unit 2154
Filing Date: 03/30/01	Docket No. 33012/312/101
For: METHOD AND MECHANISM FOR THE DEVELOPMENT AND IMPLEMENTATION OF A WEB-BASED USER INTERFACE	**SUPPLEMENTAL DECLARATION UNDER 37 C.F.R. 1.131**

Commissioner for Patents
Box 1450
Alexandria, VA 22313

Dear Sir:

CERTIFICATE UNDER 37 C.F.R. 1.8: I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to the Assistant Commissioner for Patents, Box 1450, Alexandria, VA 22313 on this 6th day of May, 2008.

By Carolyn A. Erickson

SUPPLEMENTAL DECLARATION

The undersigned declares as follows:

1. I am one of the co-inventors of the subject U.S. Patent Application, U.S. Serial No. 09/822,675, filed March 30, 2001, the co-inventors being Barbara A. Christensen, Douglas I. Langfield, Jo A. Newton, and Mary Kay K. Craig;

2. The respective home addresses of the co-inventors are:

Barbara A. Christensen
6520 White Oak Road
Lino Lakes, Minnesota 55038

Douglas I. Langfield
4251 Boulder Ridge Point
Eagan, Minnesota 55122

Jo A. Newton
1359 Michelle Drive
Eagan, Minnesota 55123

Mary Kay K. Craig
6491 White Oak Road
Lino Lakes, Minnesota 55038

3. I was employed by Unisys Corporation, assignee of the of the entire right, title, and interest of the subject invention at the time of conception and actual reduction to practice of the subject invention;

4. The invention of pending claims 1-20 of the subject U.S. Patent Application was first commercially embodied in a product of Unisys Corporation entitled Cool ICE Revision 2.1 establishing that the invention of pending claims 1-20 of the subject U.S. Patent Application was completely reduced to practice at least as early as the first commercial sale of Cool ICE Revision 2.1;

5. Cool ICE Revision 2.1 was completed before April 1, 2000 and first placed on commercial sale on April 4, 2000 as a matter of public record;

6. Attached hereto as Exhibit A is a copy of Document 7850 2473-001 published by Unisys Corporation in April 2000, entitled "Cool ICE Technical Overview", establishing that Cool ICE Revision 2.1 was on-sale as of April 2000;

7. The document of Exhibit A in general and sections 2, 4, 5, and 9-12, in particular, establish that the commercial product described therein (i.e., Cool ICE Revision 2.1), which was on-sale in April 2000, contains all of the elements and the combination thereof constituting the invention of claims 1-20 of the subject application;

8. Further declarants sayeth not.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on informaticn and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may

jeopardize the validity of the application or any patent issued thereon, I further declare that I understand the content of this declaration.

Date 6/12/07

Barbara A. Christensen

Date 6.8.2007

Mary Kay K. Craig

Mary Kay K. Craig

Date 6/8/2007

Douglas I. Langfield

5

Date 6-8-2007

Jo A. Newton

OIPE
MAY 13 2008
IAP22
PATENT & TRADEMARK OFFICE

P A T E N T

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

re application of	
Barbara A. Christensen et al.	Examiner J. Hu
Serial No. 09/822,675	Group Art Unit 2154
Filing Date: 03/30/01	
	Docket No. 33012/312/101
For: METHOD AND MECHANISM FOR THE DEVELOPMENT AND IMPLEMENTATION OF A WEB-BASED USER INTERFACE	**AMENDMENT**

Mail Stop AF
Commissioner for Patents
Box 1450
Alexandria, VA 22313

Dear Sir:

CERTIFICATE UNDER 37 C.F.R. 1.8: I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to the: Box AF, Assistant Commissioner for Patents, Box 1450, Alexandria, VA, 22313 on this 14 day of August, 20 06.

By _____
Carolyn I. Erickson

DECLARATION

The undersigned declares as follows:

1. I am one of the co-inventors of the subject U.S. Patent Application, U.S. Serial No. 09/727,607, filed December 1, 2000, the co-inventors being Barbara A. Christensen, Douglas I. Langfield, Jo A. Newton, and Mary Kay K. Craig;

2. The respective home addresses of the co-inventors are:

Barbara A. Christensen
6520 White Oak Road
Lino Lakes, Minnesota 55038

Douglas I. Langfield
4251 Boulder Ridge Point
Eagan, Minnesota 55122

Jo A. Newton
1359 Michelle Drive
Eagan, Minnesota 55123

Mary Kay K. Craig
6491 White Oak Road
Lino Lakes, Minnesota 55038

3. I am employed by Unisys Corporation, assignee of the of the entire right, title, and interest of the subject invention;

4. The invention of pending claims 1-20 of the subject U.S. Patent Application was first commercially embodied in a product of Unisys Corporation entitled Cool ICE Revision 2.1;

5. Cool ICE Revision 2.1 was completed before April 1, 2000 and first placed on commercial sale on April 4, 2000;

6. Further declarants sayeth not.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon, I further declare that I understand the content of this declaration.

Date 7/27/06

Barbara A. Christensen

Date 7/28/2006

Douglas I. Langfield

Date 8/2/2006

Jo A. Newton

Date 7/27/2006

Mary Kay K. Craig

COOL ICE

Technical Overview

UNISYS

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product or related information described herein is only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Notice to Government End Users. The software and accompanying documentation are delivered and licensed as "commercial computer software" and "commercial computer software documentation" as those terms are used in 48 C.F.R. § 12.212 and 48 C.F.R. § 227.7202-1 through 227.7202-4, as applicable. The Government shall receive only those rights provided in the standard commercial software license, or where applicable, the restricted and limited rights provisions of the contract FAR or DFARS (or equivalent agency) clause.

Correspondence regarding this publication can be e-mailed to **doc@unisys.com**

Copyright © 2000 Unisys Corporation
All rights reserved.
Unisys and Cool ICE are registered trademarks of Unisys Corporation.

April 2000

Printed in USA
Priced Item 7850 2473–001

Unisys, Cool ICE, and MAPPER are registered trademarks of Unisys Corporation.
All other terms mentioned in this document that are known to be trademarks or service marks have been appropriately capitalized. Unisys Corporation cannot attest to the accuracy of this information. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Contents

About This Overview

Purpose

This Technical Overview describes the concepts and components related to Cool ICE, and a development process for building Web applications with Cool ICE. It also provides a framework for understanding how the individual components relate to each other.

Scope

This overview describes the concepts related to Cool ICE and the purpose and function of the individual Cool ICE components. It does not contain specific procedures for using the individual components, but it does point the reader to the document where those procedures reside. The Technical Overview also contains detailed descriptions of any concepts the audience needs to understand before creating Web applications with Cool ICE.

Audience

The primary audience for this overview is Cool ICE administrators and developers. A secondary audience is IT managers or anyone else who wants a more detailed description of Cool ICE.

Prerequisites

The audience should be familiar with the following technologies:

- Windows NT operating environment
- Web server software such as Microsoft Internet Information Server (IIS), Netscape FastTrack Server, or iPlanet Web Server, Enterprise Edition
- Active Server Pages (ASP)
- ActiveX Data Object

- An HTML authoring tool such as Microsoft FrontPage or Macromedia Dreamweaver
- Scripting language such as VBScript or a Java scripting language

How to Use This Overview

Use this guide to gain a basic understanding of Cool ICE, and the individual software components of Cool ICE and how they relate to each other.

Related Product Information

This overview is part of the Cool ICE library. The following documentation is available to help you use Cool ICE. The way you use the software determines which documents you need. For example, to create Cool ICE services, read the *Cool ICE Developer's Reference*.

Printed Documentation

The following document is available in printed form:

***Cool ICE Getting Started* (7850 2481)**

This guide provides high-level installation, configuration, and overview information. It describes the Cool ICE environment, product considerations, and support information.

Online Documentation

Cool ICE also provides documentation in electronic form on CD-ROM. This documentation is in the form of either Windows Help or HTML format. You can install the documentation on a Windows workstation or on the server and view these documents online.

***Cool ICE Developer's Reference* (7850 2465)**

This reference contains the information needed to create Cool ICE services, including native Cool ICE scripted services, ASP scripted services, or data services. It also includes procedures and examples.

***Command Reference* (7850 2754)**

This reference lists the available Cool ICE commands, in alphabetical order, that you can use to write native Cool ICE scripts. It includes formats, field descriptions, and examples.

***MRI Administration and User's Guide* (7846 0391)**

This guide explains how to administer and use the MRI software. It describes the MRI and its component software and describes how to access relational databases through the interface.

***MAPPER System for Windows NT Administration Guide* (7846 0284)**

This guide provides direction for the repository administrator. It includes information about system configuration, database management, user and run registration, and configuration of communications links with other MAPPER systems.

Section 1
Introduction to Cool ICE

This section introduces Cool ICE software and describes its capabilities. It discusses the following topics:

- What is Cool ICE?
- Integrating data
- Manipulating data
- Controlling access
- Administering and managing new Web applications

What is Cool ICE?

Cool ICE is a complete environment for creating, organizing, and managing Internet and intranet applications. It helps application developers create new Web-based applications using an enterprise's existing data sources and applications. It also helps a developer or administrator organize and manage the enterprise's suite of Web applications built with Cool ICE.

Creating New Web Applications

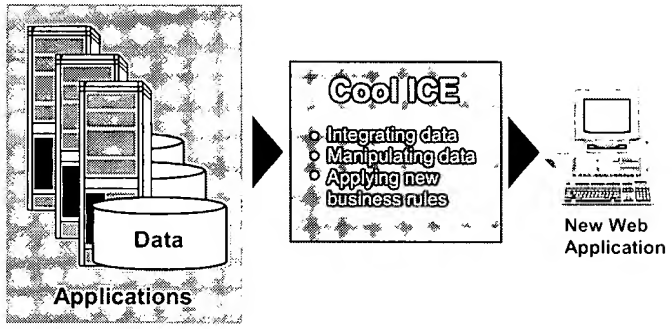
Cool ICE is designed for Web application developers. Using Cool ICE, developers create new Web applications by integrating and manipulating the data in the enterprise's existing databases and applications (such as an online transaction processing application). These new applications extend the capabilities of the enterprise's existing data sources beyond their original intent. When creating new Web applications, developers can use Cool ICE to

- Integrate data from multiple, diverse data sources. The data sources can reside on the organization's existing enterprise or departmental servers.
- Manipulate or further process the data as a single data set.
- Apply business rules or logic to the data.

Cool ICE enables Web application developers to transform the data in the enterprise's existing databases and applications by making it available for new uses and to new, Web-based users.

Illustration—Data Transformation with Cool ICE

The following figure illustrates how developers create new Web applications by using Cool ICE to transform the data in the enterprise's existing data sources.



Existing Databases and Applications

Organizing and Managing New Web Applications

Cool ICE helps developers and administrators organize and manage all their enterprise's back-end services and front-end Web applications. Cool ICE is based on its own repository. By storing all scripts, services, images, and other objects related to Web applications in the Cool ICE repository, Cool ICE enables the developer to easily organize, manage, and distribute the enterprise's Web applications. Using the administrative features of Cool ICE, developers can

- Control access to the organization's Web applications and services.
- Administer and manage the organization's suite of new Web applications.

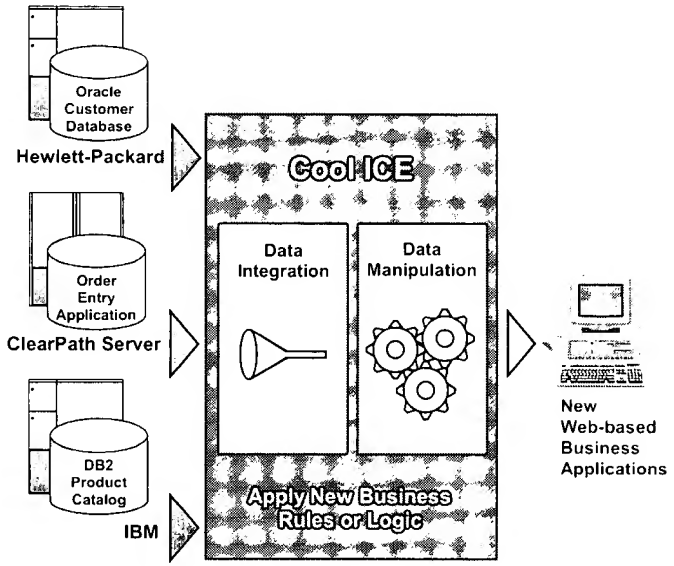
Integrating Data

Cool ICE integrates the data in an enterprise's existing, and frequently diverse, back-end databases and applications. By using the data and application access functions of the Cool ICE Engine, developers can create new Web applications that

- Access multiple, diverse, back-end databases and applications running on the existing departmental and enterprise servers. See "Data Sources" in Section 13 for the list of local and remote databases Cool ICE supports.
- Extract data from these databases and applications.
- Merge the data into a single data set.

The existing back-end databases and applications continue to support the current business processes and users. External users (those using a Web browser) always see current information from the databases since the existing applications and business processes continue to maintain the data. Any updates an external user enters into the databases are also visible to internal applications and business processes.

The following figure illustrates the data integration and manipulation capability of Cool ICE.



Manipulating Data

Cool ICE goes beyond simply Web-enabling existing databases and applications. It enables developers to further manipulate or process the data once the data is extracted from the databases and applications. Cool ICE contains two methods of manipulating data:

- Cool ICE Engine
- Data Wizard

Cool ICE Information Processing Engine

Using the analytical functions of the Cool ICE Engine, developers can perform the following types of operations on data:

- Search—find a character string
- Sort—place the lines of data in a specified order
- Count—analyze and summarize data
- Total—perform arithmetic operations on data
- Calculate—compute, compare, and replace numeric data, character strings, dates, and times
- Other analytical functions as defined in the *Command Reference*

After extracting and manipulating the data, the developer

- Adds whatever HTML and client-side scripting needed to properly display the data in a Web browser
- Charts or plots the data, if appropriate, using the Cool ICE Graphics Server

Data Wizard

The Data Wizard is a Web-based tool that provides the developer with a simple, interactive way to build application components. An application component is a self-contained script that

- Gathers data from up to 5 data sources
- Manipulates the resulting data as a table
- Formats this data as a form, table, or graph
- Reformats your data for display

For data from a single source, a developer can create an application component that updates, deletes, or inserts a single record into a database.

After creating and saving the application component, the Data Wizard builds a Cool ICE data service and a component definition. A developer can include the component definition in an active server page.

See Section 10 for a detailed description of the Data Wizard's capabilities.

Controlling Access

Cool ICE enables developers to control which users or groups of users can access the enterprise's Web applications and services. To control access, Cool ICE implements security profiles. A profile specifies which objects in the Cool ICE repository—such as services, applications, categories, images and so on—an individual user or a group of users can access. A developer or administrator allocates a profile to any object in the Cool ICE repository to which the enterprise wants to control access.

How Cool ICE Uses Security Profiles

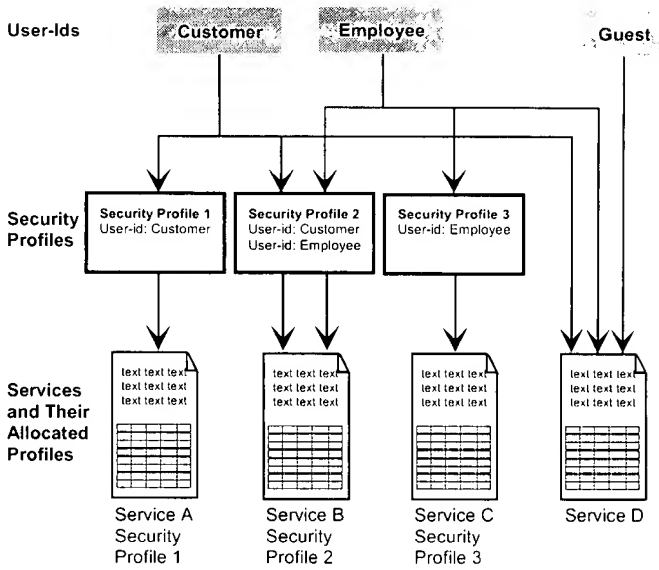
When a user requests a service, Cool ICE checks for a security profile. If Cool ICE

- Finds a security profile—Cool ICE verifies that the user is included in the profile. If the user is, he or she can execute the service. If the user is not, Cool ICE denies access to the service.
- Does not find a security profile—Cool ICE lets the user execute the service. Services without a security profile are considered "open" because any user can execute them.

In addition, with Cool ICE and security profiles, an organization needs only a single Uniform Resource Locator (URL). When a user specifies the URL for the Cool ICE gateway, Cool ICE checks the profiles and builds a menu containing only those applications and services that the user has permission to access.

Illustration—Cool ICE Access Control

The following figure illustrates the access control feature of Cool ICE.



Administering and Managing Web Applications

Cool ICE contains features to help you administer and manage your Web applications. These features include

- Cool ICE Administration Tool
- Central repository for objects
- Gateway Configuration Tool

Cool ICE Administration Tool

Cool ICE Administration Tool (ICE Admin) is one of the tools used to administer Web applications developed with Cool ICE. ICE Admin provides a point-and-click environment for maintaining services and their categories in the Cool ICE repository. Administrators and developers use ICE Admin to develop, maintain, enhance, and monitor Cool ICE services and applications.

Data Source Management

Using ICE Admin, administrators and developers can register the databases that will be available through Cool ICE, add security profiles to the databases, and maintain services in the repository.

Site Usage Information

Using the Event Viewer feature of ICE Admin, administrators and developers can view events logged by Cool ICE. This includes an access log that lists all requests for services made from a browser.

Central Repository for Objects

Cool ICE manages all objects related to a Web application—services, images, pages, files, and so on—in a common repository. Cool ICE, through ICE Admin, registers the objects in the repository and maintains the security profiles on them.

The repository resides on the local Web server, although services can be distributed across multiple servers.

Gateway Configuration Tool

The Gateway Configuration Tool is a Java applet that allows an administrator to configure how users access Cool ICE. Using this tool, the administrator configures the gateways, configurations, and connection pools that control user sessions.

Section 2
Key Concepts

This section describes some of the key concepts of Cool ICE and Web-based applications. These concepts include

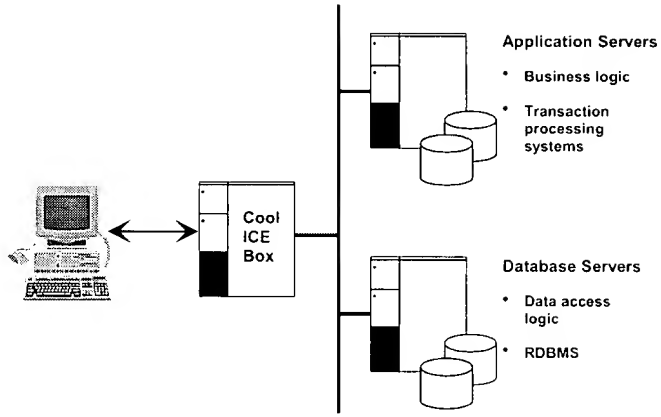
- Three-tier architecture
- Services and categories
- Session control
- State management
- Runtime flow
- Active Server Pages
- URL encoding in Cool ICE

Three-Tier Architecture

Cool ICE is typically implemented on a three-tier, distributed architecture. For some simple applications, an enterprise can implement Cool ICE on a two-tier architecture. In that case, both the business logic and database reside on the same server.

Illustration—Three-Tier Architecture

The following figure illustrates the three-tier architecture with Cool ICE executing on the middle tier or Web server.



Client Tier

In Cool ICE, the client tier is a Web browser. The browser functions as the user interface to a Web application and is responsible for handling the presentation logic. This includes displaying HTML documents and executing any client-side scripts or software components. A user executing a Cool ICE application or service can use any browser as long as that browser supports frames.

The client communicates with the Web server over a TCP/IP network using the HTTP protocol.

Middle or Process Management Tier

The middle or process management tier handles the business logic of the Web application. It is responsible for receiving client requests, handling requests for data, applying business logic to the data, and generating a client response. If the Web application is partitioned, the business and data access logic can be distributed across multiple servers in the enterprise's network.

For Cool ICE, the middle tier is a Web server executing the Windows NT Server 4.0 operating environment with either Microsoft or Netscape Web server software. The middle tier can be one of the following types of servers:

- Existing Web server
- New Web server deployed for a specific application
- Intel node of a Unisys ClearPath HMP server
- Another server in the management information systems (MIS) domain

The Web server communicates with a back-end relational database management system (RDBMS) using standard database protocols.

Cool ICE software and applications built with Cool ICE execute on the Web server. This is called the Cool ICE domain.

Data Management Tier

The departmental and enterprise servers, with their existing databases and applications, make up the data management tier. These data servers execute the Windows NT, UNIX, OS 2200, or other operating systems.

This tier can contain both application and database servers:

- Application servers handle part of the business logic and execute transaction processing monitors (TPM) such as Open/OLTP, Tuxedo, CICS, and MTS.
- Database servers handle the data access logic and provide access to external data managed by relational database management systems (RDBMS). The external data can reside in Microsoft SQL*Server, Oracle, Informix, Sybase, Unisys, and any ODBC-compliant database including DB2.

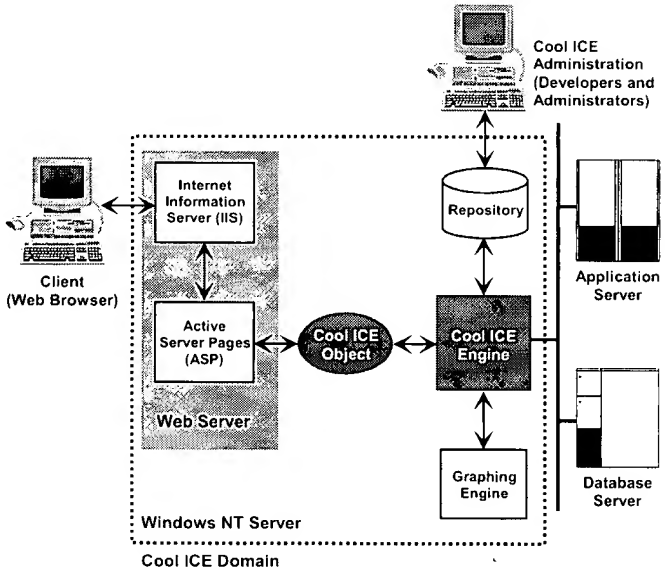
Cool ICE Domain Architecture Detail

The Cool ICE domain consists of three environments:

- Development
- Runtime
- Management

Illustration—Cool ICE Domain

The following figure illustrates the architecture of the Cool ICE domain on the Web server. The development, runtime, and management environments of Cool ICE use the components in the Cool ICE domain to create, organize, and manage the Web applications. Although the illustration shows Microsoft Internet Information Server (IIS), the Web server can also execute Netscape Web server software.



Development Environment

In the Cool ICE development environment, developers create new Web applications using Cool ICE software components and other commercial off-the-shelf software. Cool ICE contains two development environments: open and native.

Open Development Environment

In the open development environment, developers create open services using HTML and server-side scripting. An open service is an active server page that contains any combination of HTML and script logic, written with the Visual Basic or a Java scripting language (Microsoft JScript). Open services execute in the Web server environment.

Native Development Environment

In the native development environment, developers create native services composed of HTML and Cool ICE Script. Native services execute in the Cool ICE Engine. They generate HTML which is sent to the client's browser.

Development Environment Characteristics

The Cool ICE development environments have the following characteristics:

- Use standard Web authoring tools and programming environments
- Call on the Cool ICE information processing engine (Cool ICE Engine) for access to the back-end databases and applications
- Use the Cool ICE Graphics Server to produce a graphs or charts of data
- Build common application functions as reusable services
- Are based in a repository
- Through ICE Admin, allow developers to deploy services

Developers can distribute new Web applications across multiple Cool ICE servers.

Runtime Environment

In the Cool ICE runtime environment, users access the new Web applications from their browser. The Cool ICE runtime environment has the following characteristics:

- Supports standard browsers and protocols
- Supports client-side scripting and state-oriented connections
- Controls access to services through security profiles
- Accesses the existing back-end databases and applications
- Processes data locally to produce the required displays
- Supports service calls to transaction programs

Management Environment

In the Cool ICE management environment, an administrator or developer uses the Cool ICE administration component, ICE Admin, to organize and manage the new Web applications. The Cool ICE management environment

- Organizes and manages back-end services and front-end applications
- Creates and maintains the security profiles
- Controls access to applications, services, and back-end data and applications

Services

A service is a script that implements the business logic of a Cool ICE application. It performs a specific task for the end user.

Services contain any combination of HTML and scripting. A static service contains HTML and client-side scripting. Static services display text, such as product descriptions, images, forms, or any other information that does not change frequently. They return the same HTML page each time they are called by a browser.

A dynamic service contains HTML, client-side scripting, and server-side scripting. A dynamic service builds a new HTML page each time a browser calls it. The script commands in the service determine what data is processed and what work the service performs.

Types of Services

Cool ICE contains three types of services:

- Open

  An open service is an active server page that calls the Cool ICE Object. The Cool ICE Object allows the ASP page to establish a connection with the Cool ICE Engine and execute native or data services. The ASP page contains any combination of HTML, server-side scripting, and calls to the Cool ICE Object. In an open service, the server-side scripting languages are VBScript or a Java scripting language (Microsoft JScript).

  Open services execute in the Web server environment and return a formatted HTML page to the browser.

- Native

  A native service is a script written using Cool ICE Script, the native scripting language of the Cool ICE Engine. The script contains any combination of HTML and Cool ICE Script. Cool ICE stores the attributes of a native service, along with its specification, in the repository.

  Native services execute in the Cool ICE Engine and return a formatted HTML page to the browser.

- Data

  A data service retrieves and manipulates a specified data set from internal and external databases. Data services are created with the Data Wizard, one of the Cool ICE development tools. Using the Data Wizard, a developer builds a component definition. The developer can save the definition as a Cool ICE data service, or as a component definition that can be included in an active server page.

  Data services execute in the Cool ICE Engine.

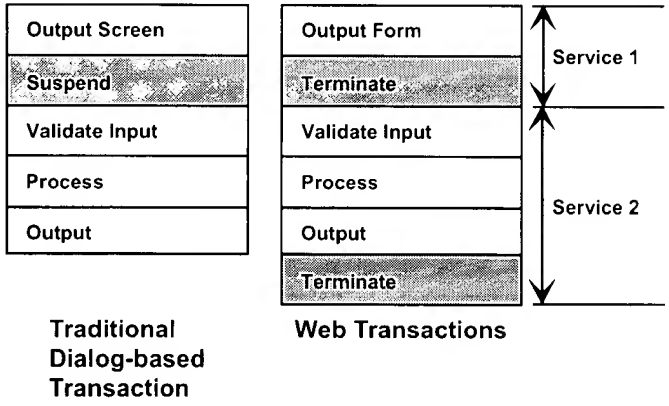
Traditional Versus Web-Based Services

The design and architecture of a Web-based service differ from a traditional dialog-based transaction in a client/server environment. A typical transaction in a dialog-based application consists of input, process, and edit. The application performs one or more transactions in a single connection with the user.

A Web service, or transaction, is stateless; that is, each Web service is a single connection to a browser and performs either input and process or process and output. In a Web transaction, you need two services to duplicate what you can perform in one dialog-based transaction. One service displays the form and another service receives and processes the input. A Web service loses its state when it terminates.

Cool ICE maintains what looks like a traditional application connection by assigning and maintaining a session identifier when a user connects through a browser. See "Session Control" later in this section for more information on the session identifier.

The following figure illustrates the differences between traditional dialog-based transactions and Web-based transactions.



Categories

Cool ICE groups services into categories. A category is a set of services that either create an application or form some other relationship. Using ICE Admin, the developer creates the categories and determines which services reside in the categories.

Cool ICE allocates a storage area in the repository to each category. This storage area contains all objects related to the services in that category.

Local and Remote Categories

A category can be either local or remote.

- Local—All services reside on the local server.
- Remote—All services reside on a remote server. Although the services physically reside on the remote server, Cool ICE maintains all information related to the remote category in the repository on the local server. ICE Admin handles the networking needed to store and retrieve services in remote categories.

Cool ICE automatically creates the drawer for local categories. For remote categories, an administrator on the remote server must manually create the drawer.

ICEAdm Category

The ICEAdm category in the Cool ICE repository contains services used by the Cool ICE software. A security profile has been allocated to some of the services. If a service

- Has the ICE-Admin profile allocated to it, the service is intended only for the person who administers the Cool ICE system. You can add additional registered users to the security profile.
- Has the ICE-Develop profile allocated to it, the service is intended only for the person who develops Web applications with Cool ICE. You can add additional registered users to the security profile.
- Does not have a profile allocated to it, the service must remain available to all users. Do not allocate a security profile to it.

See Also

See the topics related to allocating security profiles and securing Cool ICE system services in the *Cool ICE Getting Started* and *Cool ICE Developer's Reference*.

Session Control

Unlike normal applications, Web transactions are "stateless." This means that all requests to a Web server are treated as completely separate transactions. After each transaction, the state of the transaction is lost.

Traditionally, developers used one of two methods to pass information from one dialog to the next:

- Using hidden fields in forms.
- Attaching parameters to URLs.

Cool ICE, however, uses another method, a session-id, to maintain a session with the user. The session-id is a number that Cool ICE allocates the first time a user requests a service. Using cookies, the session-id is returned to the Cool ICE Service Handler each time the user accesses a Cool ICE service.

By using a session-id, Cool ICE keeps track of users and the various services and pages they request without requiring the user to supply a user-id and password each time.

State Management

Cool ICE provides subroutines and server-side variables that allow state information to be maintained and managed across multiple Cool ICE services within a session.

Subroutines

Context and state is managed by a service that saves information in a permanent report (data set) and inserts an entry into the state management table. Cool ICE provides subroutines that save, restore, and remove the report from the repository.

Server-side Variables

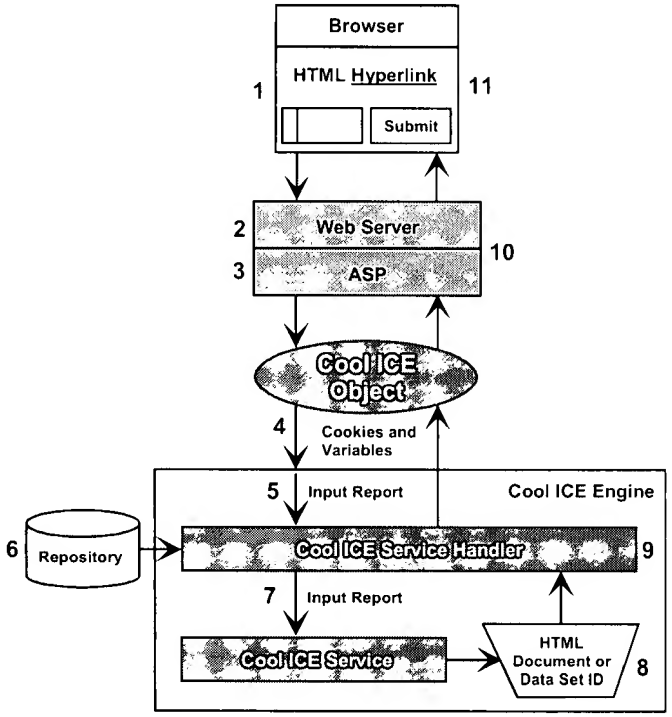
Cool ICE maintains server-side variables that allow state information to be managed across multiple Cool ICE services within a session. These variables can be called from Cool ICE services. Rather than using hidden fields or cookies to pass values from one service to another, developers can use these variables to store the values on the server and not pass them over the network.

See Also

Refer to the *Cool ICE Developer's Reference* for more information on the subroutines and server-side variables.

Runtime Flow

The following figure illustrates the Cool ICE runtime flow. In this example, a Cool ICE open service calls the Cool ICE Object to execute a native service. A description of the numbered steps follows the illustration.



The following description explains the numbered steps in the Cool ICE runtime flow.

1. The user requests a service.
2. The Web server executes the active server page, that is, the Cool ICE open service.
3. The Cool ICE open service establishes a connection and calls a Cool ICE native service through the Cool ICE Object.
4. The Cool ICE Object passes any cookies, Web server variables, and form input variables to the Cool ICE Engine.
5. The Cool ICE Engine creates a service input report from the cookies, Web server variables, and form input variables and passes it to the Cool ICE Service Handler.
6. The service handler locates the Cool ICE native service in the repository.
7. The service handler passes the input report to the service.
8. The service executes and creates either an HTML document or a data set.
9. The service handler passes the HTML document or data set ID back to the Cool ICE Object. The service handler returns control to the Cool ICE open service (active server page) through the Cool ICE Object.
10. The Cool ICE open service returns control to the Web server. The Web server passes the HTML document back to the browser.
11. The browser displays the HTML document.

Active Server Pages

Cool ICE services use Microsoft Active Server Pages (ASP). ASP is the server-side execution environment of Microsoft Internet Information Server. You can combine the following components in an ASP to create dynamic Web pages:

- HTML
- One or more scripting languages, such as VBScript or a Java scripting language (Microsoft JScript)
- ActiveX server components, such as ActiveX Data Objects (ADO)
- Calls to the Cool ICE Object

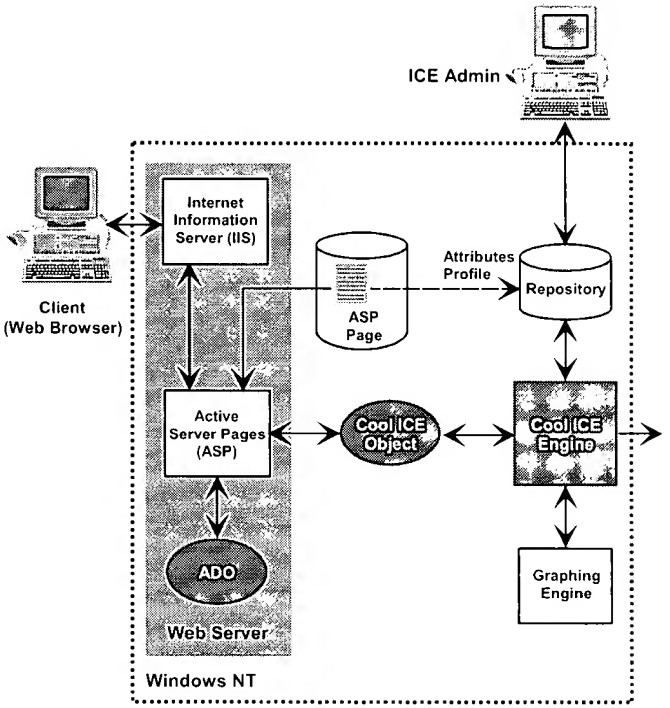
When a user requests an ASP page, the Web server executes the scripts and components on the page, formats the HTML, and sends the formatted HTML back to the browser.

ASP pages have an ".asp" extension.

If you use Web servers from Netscape Communications Corp., Chili!Soft ASP is the functional and syntactical equivalent of the ASP environment in Microsoft Internet Information Server.

Illustration—Cool ICE and Active Server Pages

The following figure illustrates the relation between Cool ICE open services (an active server page) and the Microsoft server-side execution environment. Although the illustration shows Microsoft Internet Information Server (IIS), the Web server can also execute Netscape Web server software.



ASP Templates

The Examples category includes two ASP templates: aspvb.tmp and aspjs.tmp.

aspvb.tmp

This ASP template is based on VBScript and is intended for those developers who are familiar with the Visual Basic scripting language.

aspjs.tmp

This ASP template is based on JScript and is intended for those developers who are familiar with the Microsoft JScript scripting language.

Scripting Environment

Microsoft Internet Information Server includes scripting engines for Microsoft Visual Basic Scripting Edition (VBScript) and Microsoft JScript scripting languages. Typically, these are the scripting languages used in ASP pages.

VBScript

Visual Basic Scripting Edition (VBScript) allows developers to write scripts using a subset of Microsoft Visual Basic Programming Language. It enables developers to create more interactive Web pages. Only Internet Explorer supports VBScript.

JScript

JScript is the Microsoft implementation of the European Computer Manufacturer's Association (ECMA) standard for scripting languages (ECMA-262). Its syntax is similar to C or C++. JScript allows developers to link and automate a variety of objects in HTML pages, including Java applets and ActiveX controls.

JScript Versus JavaScript

Jscript is the Microsoft implementation of the European Computer Manufacturer's Association (ECMA) standard for scripting languages, ECMA-262. This standard is based on joint submissions from both Microsoft Corporation and Netscape Communications Corporation. JavaScript is a scripting language written by Netscape that preceded the ECMA standard. JScript and JavaScript are not 100% compatible.

Virtual Directories

ASP pages are stored in virtual directories. A virtual directory is an alias for a hard-coded path on the server. Virtual directories allow you to easily move Web pages from one server to another without changing any code in the pages. They also help prevent a user from determining the physical location of the files and directories on your server.

At installation, Cool ICE creates a default virtual directory. You can either store your ASP pages in this directory or you can create your own virtual directories. Using ICE Admin, you can specify the alias and the path for your virtual directories.

Cool ICE also creates a second virtual directory for the Gateway Configuration Tool.

Cool ICE ASP Administration

ASP pages are stored as a text file outside the Cool ICE repository. For Cool ICE to organize, manage, and secure an ASP object, the developer must use ICE Admin to register the ASP page in the Cool ICE repository with its associated attributes and file location.

Inserting ASP Objects in the Repository

ICE Admin enables the developer to insert an ASP object into the repository. ICE Admin provides the following options when you insert an ASP object into the repository:

- ASP based on an ASP template. This option allows the developer to select a template from the Cool ICE repository or the default Cool ICE ASP page.
- ASP based on an existing workstation ASP script. This option allows the developer to select a file from his or her workstation that contains the ASP script. ICE Admin uploads the ASP script from the developer's workstation to the server.
- ASP based on an existing server ASP. This option allows the developer to select an ASP file from the ASP directories on the server.

Maintaining ASP Directory Aliases

ICE Admin also maintains a list of Cool ICE registered ASP directory aliases (virtual directories). When you insert an ASP page into the repository, ICE Admin allows you to specify a directory on the server where you want to store the active server pages. You select this directory from the list of ASP directory aliases you have registered with Cool ICE.

See Also

See the topics in *ICE Admin Help* related to inserting ASP objects in the Cool ICE repository.

ActiveX Data Object

The ActiveX Data Object (ADO) is one of the standard server components supplied with Active Server Pages (ASP). It provides a standard interface for OLE-DB providers. One of the most popular OLE-DB providers accesses ODBC data sources. Cool ICE provides an OLE-DB data provider which accesses data in the Cool ICE repository.

Within an ASP page you can reference both the Cool ICE Object and ADO. For example, using Cool ICE you can build a application component or a data service using the Data Wizard. Then, within the ASP you can

- Execute the data service or component definition using the ExecuteDataService method of the Cool ICE Object.
- Retrieve the data set with the RetrieveDataSet method of the Cool ICE Object.
- Manipulate the data set using the methods and properties of the Recordset object of ADO.

Chili!Soft ASP and Cool ICE

Chili!Soft ASP is a cross-platform Web application server that enables developers to build active server pages that run on Web servers from the Sun-Netscape Alliance (iPlanet) and others. It is the functional and syntactical equivalent of the ASP environment in Microsoft Internet Information Server (IIS). Chili!Soft ASP is developed by Chili!Soft, Inc. You can download a trial version of Chili!Soft ASP from the Chili!Soft home page (www.chilisoft.com).

Relationship to Cool ICE

Cool ICE supports the use of Chili!Soft ASP on Netscape FastTrack and iPlanet Web servers.

The active server pages you develop to execute in the Microsoft IIS environment should also execute, without modification, in the Chili!Soft ASP environment.

See Also

See the *Cool ICE Getting Started* for more information on using Chili!Soft ASP with Cool ICE.

URL Encoding in Cool ICE

Uniform Resource Locators (URL) in Cool ICE follow the standard syntax supported by the Microsoft Active Server Pages (ASP) environment. These URLs have the following structure. This is the syntax a developer uses for any new services he or she creates.

```
protocol://host/virtual_directory/filename?query_string
```

where

protocol:	Specifies the Internet protocol that will handle the request. This is typically the hypertext transport protocol (HTTP) but can be a Secure HTTP if Secure Sockets Layer (SSL) is being used.
host	Is either a unique machine name or the actual IP address of the site.
Virtual_ directory	Is the Active Server Pages (ASP) directory alias that is configured in Microsoft Internet Information Server and specified as the ASP directory alias in ICE Admin. Virtual_directory is also the name of the gateway that you configure using the Gateway Configuration Tool.
filename	Is the active server page that resides in the virtual directory on the host.
query_string	Is the query string used to provide data to the application. The query string consists of one or more key=value pairs separated by an ampersand (&).

Example

```
http://host/Cool-ICE/default.asp?Category=Examples&Service=FRMEX01
```

Notes

- If you do not specify the name of an ASP page, the Web server software searches the virtual directory and executes the ASP page that is configured as the default ASP page.
- If the URL does not contain a category and service name, Cool ICE displays the default category menu.

Compatibility Mode Query String Syntax

Previous releases of Cool ICE used nonstandard query string syntax. This syntax was a combination of positional values and *key=value* pairs. If you developed services using a previous release of Cool ICE, you must either set the compatibility mode flag or convert the service to the standard query string syntax. To set the compatibility mode flag, use the set_CoolICECompatibilityMode method of the Cool ICE Object. See the *Cool ICE Developer's Reference* for more information on specifying the compatibility mode flag in your active server page.

While you can use compatibility mode query string syntax with Cool ICE Release 2.0, future releases of Cool ICE may not support this syntax. To avoid future migration issues, convert any compatibility mode query string syntax to standard mode query string syntax.

System Variables Related to URLs

Cool ICE contains the following system variables to help you specify URLs. These system variables are used in the "System Variable Examples." See the *ICE Admin Help* or the *Cool ICE Developer's Reference* for a complete list of all the Cool ICE system variables.

<GURL>	Is set to equal a URL, for example, <GURL> = http://host/Cool-ICE/default.asp?
<category>	Is set to equal the name of a category; for example, <category>=IOB
<service>	Is set to equal the name of a service, for example, <service>=orderentry
<GSrvName>	Is the name of the server.

Cool ICE also includes the following system variables so the service and category syntax is compatibility with either the Cool ICE Object or the WebTx gateway. When the Cool ICE Service Handler encounters these variables, it inserts either standard or compatibility mode query string syntax, depending on whether it was called from the Cool ICE Object or the WebTx gateway.

<GCatKwd> Inserts the correct syntax to specify the category.

<GSvcKwd> Inserts the correct syntax to specify the service.

Where URLs are Used

Hypertext Links

By using the <a> HTML tag, you can link to another service:

```
<a href= "http://host/virtual_directory/my.asp"> . . . </a>
```

Form Action Field

By using the <form> HTML tag, you can specify a form and the service to call when the user submits the form:

```
<form method="POST"
action="http://host/default.asp?Category=IOB&Service=lookup">
. . .</form>
```

Section 3
Security

The Internet creates both new opportunities and new challenges for the enterprise. Using the Internet, an enterprise can improve access to information, implement new business processes, and create new models for conducting its business. The Internet is, however, a nonsecure, public network. The greatest challenge for the enterprise is to ensure that it can securely exchange sensitive data with its customers and business partners.

Application developers must address the security needs of each application they deploy on the Internet. The topics in this section briefly describe the main Internet security technologies that are available to developers. This section is divided into the following categories:

- Controlling access to systems
- Security protocols
- Cool ICE and security

Controlling Access to Systems

Access control seeks to safeguard data on the system from both unauthorized use and malicious tampering. The following topics describe two methods developers can use to control user access to systems and applications:

- User authentication (user-ids and passwords)
- Firewalls

User Authentication

User-ids and passwords are the traditional method of authenticating users and protecting systems and applications. A user-id is an identifier assigned to each person who will use the system or application. All users have their own unique user-id or share a user-id with a group of users. For example, "guest" is a typical user-id for a group of users.

A password is a character string that you can require along with a user-id to verify the use of the user-id. Applications that require a secure environment assign a unique user-id and a single password to each user.

Password Security Guidelines

The following list summarizes the guidelines you should keep in mind when using passwords to secure systems and applications:

- Remind users not to record their password in any form or reveal it to other users. Users must commit the password to memory and destroy any written evidence of it.
- Make sure users change their passwords frequently. Remind users not to reveal old passwords as it might suggest something about the general nature of the passwords they choose.
- Make sure passwords are long enough to reduce the risk of discovery by trial and error. Ideally the password should be a random combination of numbers and letters.
- Require the user to change his or her password if you suspect it has been compromised.
- Use a protected field, whenever possible, for the user to enter his or her password.

Concerns Related to Passwords

Capturing or guessing a password is typically the first method a hacker uses to gain access to a system or application. Even well-composed passwords are vulnerable to being intercepted or stolen. Because passwords are reusable, once a hacker captures a password he or she can use it over and over again.

Other Methods of User Authentication

User-ids and passwords are one form of user authentication. They rely on something the user memorizes and enters each time he or she wants access to the systems or its services. Other methods of user authentication rely on something the user has, such as a token that is generated by a smart card, or something the user physically possesses, such as a fingerprint, voice print, or thermal profile.

Advanced security techniques employ passwords, along with one of these other methods, to authenticate users.

Cool ICE and User Authentication

Using ICE Admin, the Cool ICE administration component, an administrator or developer can assign user-ids and passwords to those users who need access to the applications and services implemented under Cool ICE.

In addition to user-ids and passwords, an administrator or a developer can also create Cool ICE profiles and associate them with specific applications or services. The profile determines which applications and services a user can access.

See Also

Refer to "Cool ICE and Security" later in the section for more information on security profiles. See also the security-related topics in the *ICE Admin Help* and *Cool ICE Developer's Reference*.

Firewalls

An Internet firewall is a system designed to block unauthorized Internet users from accessing a private network connected to the Internet, typically an intranet. A firewall examines and verifies all messages entering or leaving the intranet. It blocks any messages that do not meet the specified criteria.

You do not need to include a firewall as part of your configuration if the Web application is available only on an intranet. Include a firewall if the application is available to business partners through an intranet or to customers through the Internet. You can also include both an outer and inner firewall. The outer firewall protects the front-end system from access by unintended protocols or channels, and it filters out some invalid messages. The inner firewall prevents break-ins from going beyond the front-end application to the back-end systems and databases.

Cool ICE and Firewalls

Firewalls are not intended to control access for a specific user or class of users. However, by using Cool ICE profiles, you can provide this level of control. Cool ICE enables you to maintain security profiles for

- Each registered user
- Classes of registered users
- Anonymous users

The security profile describes what portions of the application and the Cool ICE environment each user or class of users can access. Cool ICE components, along with Web server and operating system components, automatically control access to that application, service, or object. An application can also use the security profiles to customize how it processes and displays information.

See Also

Refer to "Cool ICE and Security" later in the section for more information on security profiles. See also the security-related topics in the *ICE Admin Help* and *Cool ICE Developer's Reference*.

Security Protocols

Currently, you can secure transactions over the Internet using the following protocols:

- Secure Sockets Layer (SSL)
- Secure Hypertext Transfer Protocol (S-HTTP)

SSL creates a data security layer between the application protocol, such as HTTP, and TCP/IP. S-HTTP adds message-based security specifically to HTTP. These protocols are not mutually exclusive; you can layer the S-HTTP protocol on top of SSL.

Both protocols have been submitted to the Internet Engineering Task Force (IETF) for approval as a standard.

Secure Sockets Layer

Secure Sockets Layer (SSL) is a security protocol developed by Netscape Communications Corp. It supports the following capabilities:

- Server authentication—enables a browser user to verify that messages go to the intended server.
- Message encryption—protects the entire HTTP protocol message from unauthorized reading or modifying during transmission between a server and a browser. The SSL protocol and all messages are routed through a dedicated port.
- Client authentication—enables the server to identify a specific client.

Certificates

SSL uses certificates to authenticate clients and servers. Certificates, also called digital IDs, are a string of bytes formatted according to the ANSI X.509 standard. A certificate can identify a person, system, application, or other entity.

The verification process for a certificate requires a third party, the certificate authority. The certificate authority verifies the certificates of both the Web server and the client. Third-party certificate authorities are independent companies like VeriSign.

A server certificate is required to establish an SSL session. This certificate identifies the server to the browser and contains information needed to provide secure encryption of SSL messages. Client certificates are optional. This feature of SSL enables a server to validate individual users at the start of an SSL connection.

Encryption Suites

The SSL protocol enables you to select from different encryption suites, depending on your desired level of security between the server and client. SSL supports the following encryption suites:

- Combinations of public key method (for session initiation)
- Private key method (for message encryption)
- Message digest method (for additional protection against alteration)

Encryption requires significant extra computation, so encrypting very large messages can lengthen response times and increase the system overhead.

See Also

For more information on Cool ICE and SSL, see the SSL-related topics in the *Cool ICE Developer's Reference*.

Secure Hypertext Transfer Protocol

Secure Hypertext Transfer Protocol (Secure HTTP or S-HTTP) is a security-enhanced extension to the HTTP protocol. It is designed to provide confidentiality, authenticity, integrity, and non-repudiability while supporting multiple key management mechanisms and data encryption. S-HTTP secures only specific messages (unlike SSL which secures the entire session).

S-HTTP was originally developed at Enterprise Integration Technologies with further development at Terisa Systems, Inc.

Cool ICE and Security

The following topics describe how developers can use the security profile features of Cool ICE to control access to Web applications and services:

- Levels of security
- Security profiles
- Data sources and security profiles
- Predefined security profiles
- How Cool ICE uses security profiles
- User registration

Levels of Security

Cool ICE implements two levels of security:

- Web server

  Cool ICE works with the security provided by the Web server software. A developer can use the security protocols supported by the Web browser software, such as the Secure Sockets Layer (SSL) or Secure HTTP protocols, to create secure applications.

- Application

  Cool ICE gives developers and administrators the option of adding security profiles to their applications to create an additional layer of security.

Security Profiles

Cool ICE manages access to the Cool ICE system, categories, and services using security profiles. When an administrator needs to create additional security privileges, he or she creates a security profile. A security profile consists of a list of user-ids that can access an application or service. When a security profile is complete, the administrator assigns it to one or more applications or services. Only users whose user-ids are in the security profile can then access that application or service.

To add a user to a security profile, the user's user-id must be registered. Unregistered users can access any "open" services, that is, services to which the administrator has not allocated a security profile.

In addition to applications, an administrator can associate security profiles to any object that is stored in the Cool ICE repository.

ICE Admin, the Cool ICE administration tool, is used to register users and allocate profiles.

See Also

See the *Cool ICE Developer's Reference* and *ICE Admin Help* for more information related to security profiles.

Data Sources and Security Profiles

Cool ICE allows a developer or administrator to allocate a security profile to an entire database, tables within the database, or columns within each table. Cool ICE applies the security profile in a hierarchical manner. This means a profile allocated to a database applies to all tables within the database, a profile allocated to a table applies to all columns within the table, and a profile allocated to a column applies only to that column.

The Data Wizard uses the database security feature of Cool ICE.

Predefined Security Profiles

Cool ICE contains two predefined security profiles, ICE-Admin and ICE-Develop. You can add additional user-ids to these profiles or create your own.

ICE-Admin

The ICE-Admin profile is intended for the persons who will administer Cool ICE and is used to organize and manage Web applications. After Cool ICE is installed, the ICE-Admin profile exists. However, it is not associated with any users.

By default, the ICE-Admin profile provides access to the following system services:

- Image Viewer
- Sign-on Form and Service
- Event Viewer
- Database Report Viewer

ICE-Develop

The ICE-Develop profile is intended for the persons who will develop Web applications with Cool ICE. After Cool ICE is installed, the ICE-Develop profile exists. However, it is not associated with any users.

By default, the ICE-Develop profile provides access to the following system services:

- Script Viewer
- Errors Details
- Image Viewer
- Event Viewer
- Database Report Viewer
- Data Wizard

See Also

See the *Cool ICE Developer's Reference* for more information related to the predefined security profiles.

How Cool ICE Uses Security Profiles

The following steps illustrate how the Cool ICE Object validates the user's access to an active server page.

1. The user clicks on a request for an active server page from his or her browser.
2. The Web server executes the active server page.
3. The active server page calls the ValidateAccess method in the Cool ICE Object.
4. The ValidateAccess method determines if it needs to display the sign-on form. If it does, the user enters his or her user-id and password.
5. With the information from the user, the active server page performs a profile check to see if the user is in the profile for the ASP page and has permission to execute it.
6. If the user is part of the profile, the Cool ICE Object allows the user to execute the active server page.

For native services, Cool ICE checks the profile when the user executes the service.

User Registration

The administrator or developer uses ICE Admin to register users. Registering a user specifies the user-id and password the user enters to access Cool ICE and the department to which they belong. The administrator registers only those users who need to be included in a security profile. That is, if a user needs access to a service that has a security profile, the administrator must register the user and add their user-id to the security profile.

If the administrator does not register a user, that user can still access all services that do not have a security profile associated with them (assuming the gateway is unsecured).

Departments

A department is a group of users who need similar information or perform similar processing tasks. Departments are a convenient way to group users. A number represents the department. When the administrator registers the user, he or she also assigns the user to a department.

After installation, Cool ICE contains two departments:

- Department 2

  Department 2 typically contains those users who will have administrative or development privileges. It is called the administrator's department because it contains the reports needed to administer the system. Cool ICE administrators and developers are typically registered in department 2. To execute the repository administration tool (MapAdmin), the user must be must be a member of the Windows NT administrator's group.

  The MAPNET, MAPQUE, and PACK user-ids are also registered in department 2. These are special user-ids needed to execute special Cool ICE functions and should not be deleted.

- Department 7

  Department 7 is for any other users of Cool ICE.

The Cool ICE administrator can create and register users in other departments as needed.

Mapcoord User-id

When Cool ICE is installed, a default user-id of mapcoord is registered in department 2. To preserve security, disable the mapcoord user-id once you have established a unique user-id for yourself.

Section 4
Component Summary

This section introduces the components included in Cool ICE software. It discusses the following topics:

- Development components
- Administration components
- Internal components
- Relationship of components
- Cool ICE Service Handler

Development Components

A developer uses the following Cool ICE components to create new Web applications:

- Cool ICE Object

  The Cool ICE Object is a Microsoft COM-based object that provides access to the Cool ICE Engine, Cool ICE services, and security profiles. Section 5 describes the Cool ICE Object in more detail.

- Cool ICE Repository

  The repository is a central database used to store all objects such as services, images, and so on. Section 9 provides additional information.

- Data Wizard

  The Data Wizard is a graphical user interface that provides a simple and interactive method of creating database component definitions. It is a special type of service called system-level service that is used to create services. See Section 10 for additional information.

- Script Editor

  The Script Editor is a program for editing scripts written with the Cool ICE scripting language (Cool ICE Script). See "Script Editor" in Section 7 for additional information.

Administration Components

An administrator uses the following Cool ICE components to organize and manage Web applications:

- ICE Admin

  ICE Admin, the Cool ICE administration utility, is used by administrators to manage and organize Web applications and services. Section 7 describes ICE Admin in more detail.

- Gateway Configuration Tool

  Gateway Configuration Tool is a Java applet that enables you to configure how users access a Cool ICE system. Section 8 describes this tool in more detail.

- Cool ICE Repository

  The repository is a central database used to store all objects such as services, images, and so on. Section 9 provides additional information about the repository.

Internal Components

The Cool ICE development and runtime environment also uses the following components:

- Cool ICE Service Handler

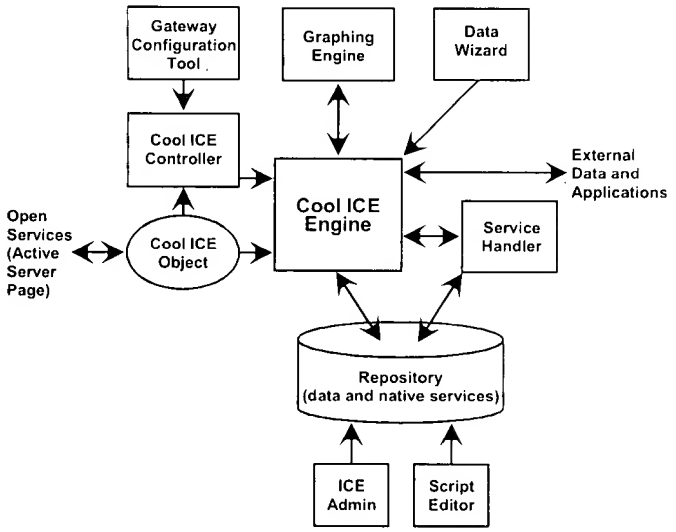
  The service handler is an internal component that determines which services are requested and ensures that the service requester has the appropriate privileges to execute the service. The service handler is a special type of service called system-level service. It is used to execute services. See "Cool ICE Service Handler" later in this section for more information about the service handler.

- Cool ICE Controller

  The Cool ICE Controller is a Windows NT service that manages Cool ICE sessions.

Relationship of Components

The following figure illustrates the relation between the Cool ICE components.



Cool ICE Service Handler

The Cool ICE Service Handler determines which services are requested and ensures that end users have the appropriate privileges to execute the requested service. It uses the Cool ICE repository to verify the service request and to help run the service. The service handler

- Processes browser requests for services
- Verifies that the user has the correct privileges to execute a service
- Maintains the service error log
- Generates a user-specific menu of categories and services
- Formats the output of a service according to the style guide

Processing Browser Requests

The Cool ICE Service Handler manages requests for services coming into the Cool ICE Engine from the browser. It delivers input from the browser to the Cool ICE Engine in a form that the engine can process. It also returns output from the service to the browser. "Formatting Service Output" describes how the service handler works with the Cool ICE style guide to format the output of a service.

Verifying Access Privileges

The service handler provides application-level security. Before Cool ICE executes a service request, the service handler verifies that the user has the necessary privileges to execute the service. If a service has a security profile, the service handler checks the profile to make sure the user's user-id is included in the profile. If it is, Cool ICE grants access to the service.

Monitoring Services

The Cool ICE Service Handler monitors user activity. This enables developers to get useful information about how services are being used.

The service handler also maintains an error log. The error log contains information about a service at the time it failed, such as the date and time of the failure, who requested the service, and internal values from the service. Using this information, developers can analyze service-related problems.

Developers can use the service handler to capture input from a browser, then step through the service in the foreground.

See Also

- For information on the error and trace logs, see Section 7, "Event Viewer."
- *ICE Admin Help* explains how to use ICE Admin to view the log files.

Generating User-Specific Menus

When a user enters Cool ICE without requesting a specific service, the service handler generates a menu that lists only those categories and services the user can access. To determine which categories and services are appropriate for that user, the service handler checks the security profiles to see which services the user can access.

Formatting Service Output

The service handler works with the Cool ICE style guide to format the output of a service. The style guide helps developers provide a consistent look and feel for HTML documents dynamically generated by a service.

The style guide also maintains common variables used in dynamically generated HTML documents. It is a single point of control for shared variables. If you change a value for a variable in the style guide, all services that reference that variable will use the new value.

ICE Admin maintains the style guide.

See Also

Refer to *ICE Admin Help* for information on using the style guide.

Section 5
Cool ICE Object

The Cool ICE Object is an object based on the Microsoft Component Object Model (COM). It provides access to the Cool ICE Engine, Cool ICE services, and security profiles from Cool ICE open services (ASP pages).

This section discusses the following topics related to the Cool ICE Object:

- Service developers and Cool ICE Object
- Properties of Cool ICE Object
- Method of Cool ICE Object
- Sample ASP page executing a native service
- Sample ASP page executing a data service

Service Developers and Cool ICE Object

Using the Cool ICE Object, developers can

- Verify that a user has the privileges to execute an active server page or native service.
- Execute a native service from an active server page.
- Execute a data service and retrieve data from an external data source and incorporate that data into the active server page as an ADO recordset.
- Upload files from a client workstation to the Cool ICE repository.
- Save state information in the Cool ICE repository and later retrieve that information.

Cool ICE maintains session information using the Cool ICE Controller. The Cool ICE Object uses a cookie to store a session identifier.

The Cool ICE Object lives as long as the execution of the active server page.

Properties of Cool ICE Object

The Cool ICE Object supports the following properties:

ProcessFormInput

Indicates that the Form Collection should be provided as input to a Cool ICE service. Its value is either true or false.

ProcessQueryString

Indicates that the QueryString Collection should be provided as input to a Cool ICE service. Its value is either true or false.

Methods of Cool ICE Object

The Cool ICE Object supports the following methods:

ValidateAccess

Determines if the user has been granted the privileges required to access this active server page. This method may return an HTML sign-on form to the browser if the configuration indicates that security is required and the user has not previously signed on. A Cool ICE security profile, created through ICE Admin, defines who can access this active server page. If the developer associated a security profile with this active server page, this method checks the profile to make sure the user's user-id is included in the profile.

ExecuteService

Executes a Cool ICE native service. A native service uses Cool ICE Script for its scripting language.

ExecuteDataService

Executes a Cool ICE data service. The Data Wizard creates a data service from a component definition. The execution of this method returns a data set ID to the ASP page. Using the data set ID and the RetrieveDataSet method, you can retrieve the data as an ActiveX Data Object (ADO) recordset.

CloseSession

Explicitly closes a Cool ICE session. This method is typically used as the action for a sign-off button.

CheckProfile

Determines if a user has the privileges to execute a specific category and service.

CreateDataSet and CreateDataSetEx

Creates an ADO recordset and saves it in the Cool ICE repository. This method provides the capability to maintain state information between transactions. It is typically used in Web applications that need to preserve information across a series of Web interactions, such as a shopping cart application.

RetrieveDataSet and RetrieveDataSetEx

Retrieves a data set previously saved in the Cool ICE repository. The data set was previously created by a Cool ICE data service or the CreateDataSet method.

RemoveDataSet

Removes a data set previously saved in the Cool ICE repository.

set_CoolICECompatibilityMode

Sets an internal property that allows the active server page to call Cool ICE 1.x native services with the ExecuteService method.

set_ProcessFormInput

Indicates that the Form Collection should be provided as input to a Cool ICE service. (This method is for compatibility with earlier releases of Cool ICE. In Cool ICE 2.1, use the ProcessFormInput property.)

set_ProcessQueryString

Indicates that the QueryString Collection should be provided as input to a Cool ICE service. (This method is for compatibility with earlier releases of Cool ICE. In Cool ICE 2.1, use the ProcessQueryString property.)

Sample ASP Page Executing a Native Service

The following ASP page executes a native Cool ICE service. It shows the scripting commands needed to

- Create an instance of the Cool ICE Object
- Execute the ValidateAccess method
- Execute the native service FRMEX01 in the Examples category

Notes:

- *If the ValidateAccess method determines that the user must sign on, it returns an HTML form that prompts the user for their user-id, password, and department number. This causes the ASP page to terminate. After the user provides their signon information, he or she must initiate an action again.*
- *The following sample script is available in Cool ICE as the frmex01 service in the Examples category.*

```
<%@ LANGUAGE="VBSCRIPT" %>

<%
Option Explicit  ' force variable declaration
Dim ErrNo
Dim ErrMsg

Call Main

'
' This is an example ASP for 2.0 style services (no compatibility
mode flag)
'
Sub ErrorHandler
ErrMsg = CStr(Err.Description)
Response.Write "<H1>Cool ICE Error Message.</H1>"
Response.Write "<HR>"
Response.Write "<H2>" & ErrMsg & "</H2><BR>"
Response.Write "Please report the error to the <A HREF="
Response.Write Chr(34) & "mailto://webmaster@thissite.com" & Chr(34)
Response.Write ">Webmaster</A> or on 1-612-555-1234<BR>"
Response.Write "Saying that error " & Hex(ErrNo) & " has
occurred.<BR>"
Response.Write "<P>"
```

```
Response.Write "<HR>"
End Sub

Sub Main
Const CIAccessAllowed = 1
Const CIAccessNotAllowed = 2
Const CISignOnRequired =3

Dim objCoolIce
Dim Status

Set objCoolIce = Server.CreateObject("CoolICE.CoolICE.1")
If IsObject(objCoolIce) = False Then
  ' Object wasn't created, return the standard error
  Err.Raise Err.Number, Err.Source, Err.Description
Else
  ' VBscript does not handle ON ERROR GOTO 999
  ' so error checking has to be inline
  On Error Resume Next
  Status = objCoolIce.ValidateAccess
  ErrNo = Err.Number
  If ErrNo <> 0 Then
   Call ErrorHandler
   Exit Sub
  End If
  On Error GoTo 0

  Select Case Status
  Case CIAccessAllowed     ' OK, Continue
   '
  Case CISignOnRequired     ' User needs to signon, return response
   Response.End
  Case CIAccessNotAllowed ' No access to the page
   Response.Write "Access is not allowed to this service"
   Response.End
  Case Else          ' Some other error, raise the standard error
   Err.Raise Err.Number, Err.Source, Err.Description
   Response.End
  End Select
End If

' VBscript does not handle ON ERROR GOTO 999
' so error checking has to be inline
' Note that the category, and service are hardcoded below
```



```
On Error Resume Next
objCoolICE.ExecuteService "Examples", "FRMEX01"
ErrNo = Err.Number
If ErrNo <> 0 Then
  Call ErrorHandler
  Exit Sub
End If
On Error GoTo 0

End Sub
%>
```

Sample ASP Script Executing a Data Service

The following ASP script executes a Cool ICE data service. It shows the methods of the Cool ICE Object needed to

- Create an instance of the Cool ICE Object.
- Execute the ValidateAccess method.
- Execute a data service. The component definition on the ExecuteDataService method was previously created with the Data Wizard.
- Retrieves the data set from the repository as an ADO recordset and displays the data

Note: *Refer to the* Developer's Reference *for additional information on calling a Cool ICE data service from an ASP page. Also, see the Volunteer Activity Centre application in the Activity Centre category to examine a working ASP-based application that calls data services.*

```
<%@ LANGUAGE="VBSCRIPT" %>

<%
Option Explicit ' force variable declaration
Dim ErrNo
Dim ErrMsg

Call Main

Sub ErrorHandler
ErrMsg = CStr(Err.Description)
Response.Write "<H1>Cool ICE Error Message.</H1>"
Response.Write "<HR>"
Response.Write "<H2>" & ErrMsg & "</H2><BR>"
Response.Write "Please report the error to the <A HREF="
Response.Write Chr(34) & "mailto://webmaster@thissite.com" & Chr(34)
Response.Write ">Webmaster</A> or on 1-612-555-1234<BR>"
Response.Write "Saying that error " & Hex(ErrNo) & " has
occurred.<BR>"
Response.Write "<P>"
Response.Write "<HR>"
End Sub

Sub Main
Const CIAccessAllowed = 1
Const CIAccessNotAllowed = 2
Const CISignOnRequired =3
```



```
Dim objCoolIce
Dim Status

Set objCoolIce = Server.CreateObject("CoolICE.CoolICE.1")
If IsObject(objCoolIce) = False Then
 ' Object wasn't created, return the standard error
 Err.Raise Err.Number, Err.Source, Err.Description
Else
 ' Set the Cool ICE Compatibility mode
 objCoolIce.set_CoolICECompatibilityMode True
 ' VBscript does not handle ON ERROR GOTO 999
 ' so error checking has to be inline
 On Error Resume Next
 Status = objCoolIce.ValidateAccess
 ErrNo = Err.Number
 If ErrNo <> 0 Then
  Call ErrorHandler
   Exit Sub
  End If
  On Error GoTo 0

  Select Case Status
  Case CIAccessAllowed ' OK, Continue
  '
  Case CISignOnRequired ' User needs to signon, return response
   Response.End
  Case CIAccessNotAllowed ' No access to the page
   Response.Write "Access is not allowed to this service"
   Response.End
  Case Else ' Some other error, raise the standard error
   Err.Raise Err.Number, Err.Source, Err.Description
   Response.End
  End Select
End If

' VBscript does not handle ON ERROR GOTO 999
' so error checking has to be inline
' On Error Resume Next
Dim dsid
Dim rs
Dim Field
dsid = objCoolICE.ExecuteDataService("Examples","qmrihouses")
'stop
Set rs = objCoolICE.RetrieveDataSet(dsid,"FALSE")
rs.MoveFirst %>
```

```
  <TABLE BORDER=1 COLS=<% = rs.Fields.COUNT%>>
  <TR>
   <%For Each Field In rs.Fields %>
   <TH> <% = Field.Name %> </TH>
   <% Next %>
  </TR>
<% Do While Not rs.EOF %>
<TR>
  <% For Each Field In rs.Fields %>
  <TD ALIGN=LEFT>
  <% If IsNull(Field) Then
   Response.Write " "
  Else
   Response.Write Field.Value

  End If %>
</TD>
<% Next
rs.MoveNext %>
</TR>
<% Loop %>
</TABLE>
<% rs.Close
   Set rs = Nothing
' objCoolICE.RemoveDataSet(dsid)

ErrNo = Err.Number
If ErrNo <> 0 Then
Call ErrorHandler
Exit Sub
End If
On Error GoTo 0

End Sub
%>
```

Section 6
Cool ICE Engine

The Cool ICE Engine is an information processing engine that handles tabular and free-form data sets. The information processing functions of the Cool ICE Engine are directly accessible through Cool ICE Script. Some functions are indirectly accessible through the Data Wizard.

This section discusses the following topics:

- Analytical functions
- Data access functions
- Application access functions
- Cool ICE Script

Analytical Functions

These functions perform common business data processing operations on whole data sets in a single call to the Cool ICE Engine. For example, the analytical functions enable a developer to

- Sort data on multiple keys.
- Compute multiple subtotals and totals.
- Perform global search and replace operations.
- Perform arithmetic calculations.
- Correlate several data sets and merge them into a consolidated result set.

Data Access Functions

The data access functions of the Cool ICE Engine give users access to external data managers such as ODBC, Oracle, Informix, Sybase, Ingres, and others. These functions enable developers to create services that do the following:

- Retrieve data from an external data manager and bring it into the Cool ICE environment.
- Insert, update, and delete data from tables stored in external data managers, without having to know anything about how the data is actually stored.
- Create or delete complete tables in external data managers.

The data access functions provide a seamless connection to external data. Using these functions you can access local databases, remote databases using networking between two servers running Cool ICE software, and remote databases using the database vendor's networking capabilities.

Application Access Functions

These functions access the OLTP environment and enable developers to create Open/OLTP clients and servers.

Cool ICE Script

Cool ICE Script is the native scripting language supported by the Cool ICE Engine. Using Cool ICE Script you have a stored procedure capability so that you can execute general or application-specific sequences of operations with a single call.

See Also

The *Command Reference* describes the statements in Cool ICE Script.

Section 7
Cool ICE Administration Tool

The Cool ICE Administration tool, ICE Admin, is the graphical interface used by a developer to create the services and categories for new Web applications. A developer or administrator also uses ICE Admin to administer, manage, and maintain the enterprise's Web applications.

This section discusses the following topics related to ICE Admin:

- Service development and ICE Admin
- Administration and ICE Admin
- Secure ICE Admin environment
- Database access
- Event Viewer
- Script Editor
- ICESETUP script

Service Development and ICE Admin

Developers who create Cool ICE services typically use ICE Admin to complete the following tasks:

- Create a new category for the application and specify its properties, such as drawer width.
- Register any other objects the service will use in the Cool ICE repository. This includes images, applets, text documents, and so on. To improve performance, you may want to store images on the server rather than in the Cool ICE repository. If so, use ICE Admin to register an alias for the image directory on the Web server.

- If needed, register any users who access the new service and category, create a security profile, and allocate the security profile to the service.
- Register any databases and specify the tables the service will access.

Other tasks depend on the type of service you are creating—open, native, or data.

Open Services

For open services, use ICE Admin to

- Specify an alias for the directory on the server where you will store the active server pages.
- Register the attributes of the active server page in the Cool ICE repository.

Native Services

For native services, use ICE Admin to specify a script editor—either the Script Editor, the Cool ICE report editor, or any other editor on your workstation.

Data Services

For data services, use ICE Admin to

- Register the databases, tables, and columns that will be available for developers to use in services.
- Allocate security profiles, if needed, to the databases, tables, and columns. The security profile limits access to the data sources to specific developers.
- Allocate a security profile to the service itself, once it's created.

Administration and ICE Admin

Developers or administrators who maintain the enterprise's new Web applications typically use ICE Admin to complete the following tasks:

- Rename, copy, delete, or deregister any object in the Cool ICE repository.
- Delete or deregister a category.
- Import or export a category.
- Determine the desired sign-on and system settings.
- Create a style guide. The style guide creates a consistent look and feel to the HTML documents a service generates and maintains common variables used in dynamically generated HTML documents.

Secure ICE Admin Environment

To execute ICE Admin, an administrator or developer needs a user-id and password to the Cool ICE system and membership in the ICE-Admin profile. Once an administrator or developer executes ICE Admin, they have access to all functions, categories, and objects in the repository.

A Cool ICE administrator can, however, further secure the administration environment by restricting access to

- Individual ICE Admin functions

  Profiles, which control user access to Cool ICE categories and services, can also control access to the administration environment. An administrator can allocate profiles to individual ICE Admin functions such as Create New Category, Export and Import Category, Open Object, and so on. Cool ICE dynamically generates the menu and function bars so they show only those functions whose profile matches the profile of the user.

- Categories and objects in the repository

  An administrator can also allocate profiles to the categories and objects in the repository. This means that when an ICE Admin user browses the repository to open a category, ICE Admin lists only those categories with a profile that matches the profile of the user. Similarly, when the user selects a category, ICE Admin lists only those objects with a profile that matches the profile of the user.

The ICE Admin Security function on the Security menu in ICE Admin secures access to the administration environment. Initially, the function is allocated to the ICE Admin profile. After installation of Cool ICE, the Cool ICE administrator, using the MapCoord user-id, can initially assume the role of administrator.

Database Access

Using ICE Admin, a developer or administrator can add, modify, or remove access to a database and the tables within that database. Once you register a database in Cool ICE, you can use the Data Wizard to access that database.

ICE Admin also allows you to allocate security profiles to any registered database. The Data Wizard uses the security profile to determine which databases and tables the user can access. Profiles are hierarchical. If you allocate a profile to a database, the profile restricts access to the database and all tables within the database. If you allocate a profile to a table, the profile restricts access to the table and any columns within the table.

At runtime, the Data Wizard checks to see if the user has the privileges needed to access a database or one of its tables. If the user's user-id is included in the profile allocated to the database, the Data Wizard allows the user to access the database.

Event Viewer

Cool ICE maintains access, error, and trace log files that the administrator or developer can use to monitor and maintain the Web applications built with Cool ICE. The Event Viewer menu in ICE Admin enables you to view the contents of the log files.

Access Log

In the access log, Cool ICE records all requests for services made from a browser. Each day, Cool ICE allocates a new file to contain all the service requests logged for that day.

At the first service request of a new day, Cool ICE summarizes the log data of the previous day and adds the data to a summary report. The summary report is kept in report 8F in the drawer where Cool ICE is installed. To view this report, use the Cool ICE Client (you cannot view it through ICE Admin).

Error Log

In the error log, Cool ICE records service failures due to a syntactical error in the service. To help the developer correct the error, Cool ICE also records information about the service environment at the time of the failure.

Trace Log

In the trace log, Cool ICE records information for services that have the Trace On attribute selected in the Service Attribute dialog. The View Trace Log option displays entries in the log file. Selecting a log entry from the list shows the information logged for that incident.

See Also

See the *ICE Admin Help* for more information related to the Event Viewer and the access, error, and trace logs.

Script Editor

The Script Editor is a 32-bit Windows program for editing scripts written with the Cool ICE scripting language (Cool ICE Script). Its main features include

- Full syntax color coding that each user can customize
- Ability to set bookmarks or flags
- Find and replace commands
- Undo and redo commands
- Support for Object Linking and Embedding (OLE) drag and drop

You can specify the Script Editor as your default editor when you click the Cool ICE Script icon on the Object Attributes dialog in ICE Admin.

The Script Editor is not designed to be used as a standalone editor. It is available only through ICE Admin.

See Also

- Refer to the *ICE Admin Help* to specify a workstation editor.
- Refer to the *Script Editor Help* for details on using the script editor.

ICESETUP Script

ICESETUP is a script in the Cool ICE repository that an administrator uses to

- Upgrade or reinstall the Cool ICE repository and administration components (ICE Admin and Cool ICE Service Handler).
- Create a new Cool ICE system with its own set of administration components. This enables each developer to have his or her own workspace that is totally separate from the workspaces of other developers.

To start the ICESETUP script, type ICESETUP on the top line of the Cool ICE Client and press NumEnter. See the *ICESETUP Help* for information on using the script.

Section 8
Gateway Configuration Tool

The Gateway Configuration Tool is a utility that an administrator uses to configure and manage the connections between the Web server software and the Cool ICE system.

This section discusses the following topics:

- Gateway configuration
- Default gateway configuration
- Relation between gateway and virtual directory

Gateway Configuration

Using the Gateway Configuration Tool, the administrator configures the following components:

- Connection pools—the pools of active Cool ICE connections associated with a Cool ICE site. Connections remain in the connection pool until the Cool ICE system is restarted. A connection pool contains one or more configurations.
- Configurations—the set of parameters used to establish a connection to the Cool ICE Engine. A configuration defines characteristics such as session timeout value, a default user-id and password for anonymous access to Cool ICE services, and various other parameters. A configuration contains one or more gateways.
- Gateways—the virtual directories on the Web server that contain the applications (ASP pages) that a user executes. The gateway name must match the name of the server ASP directory configured through ICE Admin.

The Gateway Configuration Tool is a graphical interface that you run from your Web browser.

Access Permissions

When you enter the Uniform Resource Locator (URL) for the Gateway Configuration Tool (http://*server-name*/CISystem/GateTool), the server prompts you for a user-id and password. The user-id you enter must be a member of the Windows NT Administrator group. Other users cannot access the Gateway Configuration Tool.

Navigation Tree

The Gateway Configuration Tool uses a navigation tree to show the relationships among the connection pools, configurations, and gateways. The following figure shows the navigation tree for the default gateway configuration released with Cool ICE.

Default Gateway Configuration

Cool ICE is released with a default connection pool, configuration, and gateway. When you develop your Web applications, you can either

- Use the default connection pool, configuration, and gateway as is.
- Modify one or more parameters in the default connection pool, configuration, or gateway.
- Create your own connection pools, configurations, or gateways.

Note: *If you create new connection pools, configurations, or gateways, do not delete the default connection pool, configuration, or gateway.*

Default Configuration Pool

The parameters in the default connection pool (Cool-ICE) have the following values. Refer to the *Gateway Configuration Tool Help* for more information about each parameter.

Max Connections	31 users
Max Wait Time	20 seconds
Site Letter	A
Character Set	Multinational

Default Configuration

The parameters in the default configuration (Cool ICE Default) have the following values. See the *Gateway Configuration Tool Help* for more information about each parameter.

Session Timeout	14 minutes
Secure SignOn	N (No)
Default User	webuser
Default Department	7
Default Password	webusr
Service Handler	ICESVCHND
SignOn Service	SIGNON
User Validation Service	USERCONTROL
Temporary Directory	C:*Cool ICE install directory*\Temp where *Cool ICE install directory* is the program folder where Cool ICE was installed. The default is cool_ice.

Default Gateway

The default Active Server Pages (ASP) file name for the default gateway (Cool-ICE) is default.asp.

Relation between Gateway and Virtual Directory

In Cool ICE, the name of a virtual directory must match the name of a gateway. Cool ICE uses the name of the virtual directory to determine which gateway to use. Once Cool ICE knows the name of the gateway, it

- Follows the hierarchy of the navigation tree to determine the configuration and connection pool associated with that gateway
- Uses the values for the parameters in the configuration and connection pool to control the user's session

If you create multiple gateways, each gateway name must be unique across all configurations and connection pools.

When Cool ICE was installed on your server, it created a default virtual directory (Cool-ICE) to correspond to the default gateway (Cool-ICE) defined in the Gateway Configuration Tool. Cool ICE also installs a default ASP page (default.asp) in the virtual directory. This enables developers to begin creating applications with minimal system configuration.

Section 9
Cool ICE Repository

The repository is a database that resides on the Windows NT server. As the central storage location for Cool ICE, it stores objects related to Cool ICE services, system programs related to Cool ICE, and state information and temporary data.

This section discusses the following topics:

- Object types
- System program storage
- State information and temporary data storage
- Repository administration tools
- Structure of the repository
- Data storage in the repository

Object Types

The Cool ICE repository stores all objects managed through ICE Admin. The repository contains predefined object types. Using ICE Admin, you can create additional object types. The following table lists the predefined object types.

Acronym	Description
ASP	Attribute profile information for an ASP page. (The repository does not store the actual ASP page.)
CDA	Data Wizard Component Definition Action
CDS	Data Wizard Component Definition Standard
CDT	Data Wizard Component Definition Transaction
CLA	Applet
DOC	Document created with Microsoft Word (*.doc file)

Acronym	Description
DYN	Dynamic service
GIF	Graphics Interchange Format image
HTM	HTML object
JPG	JPEG image
PPT	Presentation created with Microsoft PowerPoint (*.ppt file)
RPT	Cool ICE database report, that is, a report that is stored in the Cool ICE repository. A Cool ICE report is the same as a MAPPER report.
STA	Static document
TMP	Template
XLS	Spreadsheet created by Microsoft Excel (*.xls file)

Using ICE Admin, the Cool ICE administrator can associate a service with each predefined or user-defined object type. If a user selects an object with an associated service, Cool ICE starts the service and then the service processes the content of the object.

System Program Storage

The following system programs reside in the Cool ICE repository:

- Data Wizard
- ICE Admin
- Service Handler

In addition, the repository stores the examples of Cool ICE services. Because these programs and examples reside in the repository, you cannot use their location to store your own services or objects. The topic "Cabinets Related to Cool ICE" later in this section provides additional information.

State Information and Temporary Data Storage

Cool ICE services use the repository to save state information between ASP pages. A service can also use the repository as a temporary storage area for the data a service retrieves from external databases. This allows the service to use the analytical functions of the Cool ICE Engine to further process the data before returning it to the browser.

Repository Administration Tools

Cool ICE contains a set of tools an administrator can use to complete the following tasks:

- Secure the repository
- Troubleshoot
- Audit the repository
- Back up the repository
- Establish network connections
- Add a file to the repository

The main tools to administer the repository are the MAPPER Administration Program (MapAdmin) and the *MAPPER System for Windows NT Administration Guide*. The repository is based on MAPPER software, a product developed and marketed by Unisys Corporation. The tools to administer the Cool ICE repository are the same tools used to administer the MAPPER System for Windows NT.

MAPPER Administration Program

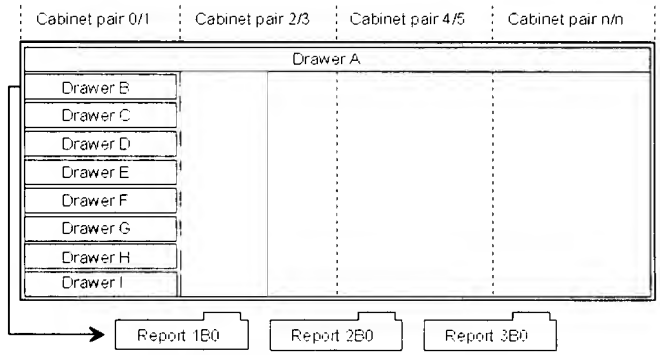
The MAPPER Administration program (MapAdmin) is the graphical interface to the administrative tasks that help you configure and maintain the Cool ICE repository. MapAdmin executes either locally on the Windows NT server or remotely from a Windows NT or Windows 95/98 workstation that has MapAdmin installed. The workstation must be connected to the server by TCP/IP.

MAPPER System for Windows NT Administration Guide

The *MAPPER System for Windows NT Administration Guide* provides direction for the Cool ICE repository administrator. It includes information about system configuration, repository management, auditing the repository, backing up the repository, and configuring communications links with other Cool ICE systems, as well as day-to-day administrative functions.

Structure of the Repository

The method of organizing information within the Cool ICE repository is similar to that of a traditional office. Think of the repository as a large storage room. Within this room are filing cabinets containing drawers. These drawers, in turn, contain information in the form of reports. Cool ICE adapts this structure to the repository. You access cabinets, drawers, and reports to recall, review, and revise the information in your reports.



Cabinets

You use sections of the repository called cabinets (so named because they can be compared to filing cabinets) to store important information.

Cabinets are numbered and matched in pairs that include an even number and an odd number, starting with cabinet pair 0/1. The repository contains 2001 cabinet pairs—cabinet pair 0/1 through 4000/4001.

Drawers

Just as a filing cabinet contains drawers, each cabinet in the repository contains eight drawers. Each drawer is a subcategory for the specific topic to which that cabinet is dedicated. For example, the cabinet for a sales department might contain one drawer for each salesperson.

Drawers are named by a letter, A through I. Drawer A is the free-form drawer, which is like a blank sheet of paper that you can use for any sort of data. Drawer A is shared by all cabinets, but it is not displayed in the drawer table of contents. It contains general information that is not protected.

Drawers B through I are your databases, laid out in either line and column format or free-form, like drawer A.

Reports

Reports are sets of related data that reside within drawers of a cabinet. Each drawer can store a large number of reports. A report can contain the objects you use in your service. To identify a report, specify a report number (2), a drawer letter (B), and a cabinet number (0); for example, 2B0.

Reports can contain up to 1,000,000 lines. The maximum number of lines per report may vary from drawer to drawer. See the "LIMITS" and "@DRW (Drawer)" topics in the *Command Reference* for more information about the maximum size of reports.

The repository administrator controls the size of reports and drawers, and assigns new drawers for new applications.

Using the Cool ICE Client, you can view the contents of the reports in the repository.

Cabinets Related to Cool ICE

Certain cabinets in the repository are used by Cool ICE. Consequently, you cannot use these cabinets to store your data or objects. The following cabinets are reserved for system use:

- Cabinet pairs 0/1 through 14/15
- Cabinet pair 320/321
- Cabinet pair 322/323

  Cabinet 322 contains examples of Cool ICE services that you can use for reference or for templates when you build your own services. This cabinet contains the following examples:

  - Drawer B—Examples of various forms such as an input form, fill-in form, and so on
  - Drawer C—Source services and objects for Cool ICE Athletic Footwear, an electronic business application
  - Drawer D—Source services and objects for the In and Out Board, a typical office application

Sites and the Site Letter

A site is a Windows NT server (the Web server) that has its own Cool ICE repository, list of users, and general operations. Each site is represented by a letter of the alphabet called the site letter. During installation, Cool ICE installs a default site with a site letter of A.

If the server has enough disk space and memory, you can set up more than one site on a server. Each site has its own repository, list of users, and general operations. For example, one user may use site A, and another user in a different group may use site B. On any given server, the site letters must be unique. Multiple sites on one server are sometimes called local sites.

There are several reasons for using multiple sites:

- You have two or more unrelated repositories that you wish to isolate for easier maintenance and management.
- You want separate sites for development, test, and production.

Using the repository administration tools, you administer each site separately.

Data Storage in the Repository

In addition to storing the objects a service uses, the repository can also store data in reports. A Web application or a service can then use the data in the reports. The topics on this page describe the information you need to understand how Cool ICE stores data in the repository.

Displaying a Report

Use one of the following methods to display a report:

- Type a report identifier that includes a report number, a drawer letter, and a cabinet number, for example, 2B0 on the top line of the Cool ICE Client.
- If the function key bar is visible at the bottom of the Cool ICE Client, press Drawer to display a list of available drawers; highlight the desired drawer (tab to it or click on it) and press the Enter key on the numeric keypad (the NumEnter key). Highlight the desired report and press the NumEnter key. If you use a laptop that does not have a NumEnter key, use Ctrl-X.

Report Screens

When you display a report, a screen similar to the following appears:

```
Line>1              Roll>              1F0
.DATE 29 SEP 98 09:51:04 RID 1F 29 SEP 98 MICRO
.FootWear - Inventory
*Item  .   .          .      .   .                                        .Unit .
*Number.Qty. Type     .Gender.Siz. Color                                   .Cost .
*======.===.==========.======.===.==================================.======.
  32330   7 Tennis     Mens   6   White/Black-Hot Lime, Neo Teal      25.00
  32331   5 Tennis     Mens   7   White/Black-Hot Lime, Neo Teal      30.00
  32332   3 Tennis     Mens   7   White/Black-Hot Lime, Neo Teal      30.00
  32333   6 Tennis     Mens   8   White/Black-Hot Lime, Neo Teal      35.00
  32334   0 Tennis     Mens   8   White/Black-Hot Lime, Neo Teal      35.00
  32335   0 Tennis     Mens   9   White/Black-Hot Lime, Neo Teal      40.00
  32336   0 Tennis     Mens   9   White/Black-Hot Lime, Neo Teal      40.00
  32337  10 Tennis     Mens   10  White/Black-Hot Lime, Neo Teal      45.00
  32338   9 Tennis     Mens   10  White/Black-Hot Lime, Neo Teal      45.00
  32339   7 Tennis     Mens   11  White/Black-Hot Lime, Neo Teal      50.00
  32470   5 Tennis     Womens 6   White/Light Marine-Ultramarine      19.70
  32471  20 Tennis     Womens 6   White/Light Marine-Ultramarine      19.70
  32472  20 Tennis     Womens 7   White/Light Marine-Ultramarine      22.39
  32473   7 Tennis     Womens 7   White/Light Marine-Ultramarine      22.39
  32474   0 Tennis     Womens 8   White/Light Marine-Ultramarine      25.00
  32475   3 Tennis     Womens 8   White/Light Marine-Ultramarine      25.00
  32476  10 Tennis     Womens 9   White/Light Marine-Ultramarine      29.72
  32477   5 Tennis     Womens 9   White/Light Marine-Ultramarine      29.72
1   2Paint  3SOE  4Return  5   6Tasks  7View   8Help  9   10Edit
```

Control Line

The first line on the screen is the control line. The control line has two fields that enable you to display different parts of the report and one field (on the far right) that denotes the report being processed. You also enter command calls on the control line. It is important to note that the control line is not a part of a report. For example, when you print or count lines in a report, the control line is not included.

Here is a typical control line:

```
Line>24 Roll> 1F0
```

The Line field indicates the number of the line currently at the top of the screen. The preceding control line indicates that line 24 of the report is at the top of the screen. To bring a different line to the top, type its number in the Line field. For example, to display line 15 at the top of the screen, type 15 in the Line field and press NumEnter.

The Roll field scrolls the display forward or backward. To scroll forward, type + or leave the field empty and press the Enter key on the numeric keypad (NumEnter key). To scroll backward, Type - and press NumEnter. To scroll forward a certain number of lines, type + and the number of lines, then press NumEnter. To scroll backward a certain number of lines, type - and the number of lines. For example, -13 scrolls the display backward 13 lines.

You can shift the displayed report in other ways, too. For example, using the Display Alternate Format (Fn) command, you can display a different selection of fields. Use the following commands to manipulate the report on the screen:

- Create Temporary Format (VIEW)
- Display Alternate Format (Fn)
- Display and Hold Headings (DH)
- Hold Characters on Screen (HCn)
- Hold Lines on Screen (Hn)
- Shift Report (Sn)

For information on these commands, see the *Command Reference*.

Function Key Bar

The bottom line of your screen is the function key bar. It contains the names of keys to press to perform commands and navigate through the repository. Like the control line, the function key bar is not considered part of a report.

```
1 2Paint 3SOE 4Return 5 6Tasks 7View 8Help 9 10Edit
```

The function keys in the function key bar vary depending on what you are doing.

The function key bar, optionally, is displayed at the bottom of the screen. If it is not, click on FKeys in the menu bar at the top of the screen to display the list of available function keys.

Function Keys

The following table describes some of the keys that you may see displayed on the function key bar. The keys available on the function key bar change to fit your environment and the application you are using.

Addlin

> Adds a line to the report where the cursor is positioned. This function key on the function key bar is available only after you press Edit.

Edit

> Brings you into edit mode. From there you have access to the Line Change menu, from which you can choose to add, delete, or duplicate lines.

LineCh

> Displays a menu from which you can change a certain line. For example, to duplicate a line, place the cursor on the line to be duplicated and press LineCh. Then move the cursor to the Duplicate Line selection and press NumEnter. This function key on the function key bar is available only after you press Edit.

Paint

> Repaints the screen. For example, with a report on display, type some text but do not press NumEnter. If you press Paint, the report is redisplayed without the text you just typed.

Remote

Lists systems to which you have access. You can transfer your station to another site by pressing Remote, moving the cursor to a system on the menu, and pressing NumEnter.

Report

Lists all the reports to which you have access. You can choose a report in which to work by pressing Report, moving the cursor to a report on the menu, and pressing NumEnter.

Resume

Continues a function from the point at which it pauses or stops. For example, since the Find (F) manual function pauses after it finds the first occurrence of a character string, you must press Resume to find successive occurrences.

Return

Displays a previous screen. For example, if you display a report while the active screen is on display, pressing Return redisplays the active screen. (The screen you are returned to varies depending on the current screen display and the path you used to obtain it.)

Runs

Lists all the runs to which you have access. You can execute one of the runs by pressing Runs, moving the cursor to one of the run names, and pressing NumEnter.

SgnOff

Brings you back to the sign-on screen. From there, you can sign on to the Cool ICE repository again or exit.

SOE

Signs you on to the Cool ICE repository as a new user. This gives you access to the demonstration database and the basic manual functions. Use this for training purposes only.

Tasks

Displays a selection of jobs you might want to accomplish, such as printing, comparing reports, or finding data. For example report, press Tasks, then select Print from that menu. Another menu then displays specific print commands from which you can choose.

Undo

Reverses the last update command performed. This key does not undo the action of a run, but it does reverse the action of some commands (for example deleting a report). Undo is usually displayed on the function key bar; if it is not displayed, you cannot undo the last update operation.

View

Displays a menu of selections that enable you to control the report display. For example, to shift the report 35 characters to the left, press View, then enter 35 in the Shift menu selection and press Enter.

Report Structure

The following figure illustrates the basic structure of a report. A description of the lines in the report follows the figure.

```
.DATE 29 SEP 98 09:51:04 RID 1F 29 SEP 98 ICE-Admin <-- Date line
.FootWear - Inventory                               <-- Report title line
*Item  .   .          .          .                   <-- Field heading lines
*Number.Qty. Type     .Gender.Siz. Color                  (two shown)
*======.===.==========.======.===.================ <-- Heading divider line
32330     7 Tennis Mens 6  White/Black-Hot Lime,    <-- Data lines
32331     5 Tennis Mens 7  White/Black-Hot Lime,        (only two shown)
```

Date Line

When a report is on display, the first line of a report is the date line, which is also called line 1. The date line looks something like this:

```
.DATE 29 SEP 98 14:17:44 RID 1F 01 JUN 98 ICE-Admin
```

Following is an explanation of each field of the date line:

.DATE	Date line identifier (also called line 1)
29 SEP 98	Date of last update
09:51:04	Time of last update
RID 1F	Report number and drawer letter
29 SEP 98	Date of origin of report or date report was last replaced
ICE-Admin	User-id of last user to update report

Heading Lines

The heading lines are below the date line and contain the following types of lines:

- The report title begins in column 2 of the line, with a period in column 1. This title is the same as the title that appears in the Report Select menu.
- Field headings begin with an asterisk (*) in column 1.
- The heading divider line separates the field headings from the data. It begins with an asterisk (*) in column 1 and then contains equal signs (=) denoting the size of each field, with periods separating each field.

Data Lines

The lines below the heading divider line contain the actual data of the report and are called data lines. The data lines are commonly structured into fields. Each field is made up of one or more columns (character positions) that are typically separated by tab characters.

Each line in the report begins with a character other than your data, such as an asterisk, a period, or a tab. These characters determine the way Cool ICE interprets each line.

Line Types

Reports can contain different types of lines, or line types, which are used to distinguish different kinds of data. For example, the line type determines whether the line is column-formatted or whether the system edits it for allowable characters. The line type is defined by the character in column 1.

If you are designing a report, use line types to specify how the system will interpret your data. Use the following descriptions to decide which line types to use. If you are using a report that someone else created, read the following descriptions of each line type to understand how the report was set up and how to use each type of line.

The leftmost character on a line determines the line type. Reports can have these kinds of line types:

- Tab—Column-formatted edited lines, starting with a tab character.
- Asterisk—Column-formatted nonedited lines, starting with an asterisk (*).
- Period—Free-form comment lines, starting with a period (.).
- Special—Column-formatted edited lines, starting with any character other than a tab, period, or asterisk.

When determining which line types to use, remember that if you use the Search (S, SRH) command, asterisk and period lines are trailer lines. This means that they are associated with the previous data line and remain with that data line when it is processed. For example, a Search result shows all tab lines found, along with any asterisk or period lines that follow each tab line. In the Search command, asterisk lines are trailer lines to tab lines, and period lines are trailer lines to asterisk lines. In other commands, the system considers all line types other than the line type being processed as trailer lines.

Tab Lines

Type a tab character in column 1 to specify a column-formatted, edited line. The tab character is commonly represented on the screen by a vertical bar (|) but may be displayed as a space or a small dot. Most data lines in reports are tab lines. Tab lines range from 40 to 998 characters in length.

You can display and process tab lines in different formats.

If you use the Cool ICE Client, pressing Ctrl-V toggles between a visible and invisible tab character.

Asterisk Lines

Type an asterisk (*) in column 1 to specify a column-formatted, nonedited line. Use asterisk lines for report headings. You can also use asterisk lines for data lines if you will use the report only with Cool ICE Script. Asterisk lines range from 40 to 998 characters in length and can be displayed and processed in different formats.

In the Search command, asterisk lines are considered trailer lines if they follow tab lines or other asterisk lines.

Note: *Do not use asterisk lines for data lines if you want to use the report with the Data Wizard.*

Period Lines

Type a period in column 1 to specify a free-form, nonedited line. The date line and title line of reports are period lines. You can also use period lines to include comments in the report, since Cool ICE does not interpret these lines as data. Period lines do not shift from left to right when you manipulate the report display.

Period lines are restricted to 80 to 132 characters in length. Period lines do not shift. So if your terminal is only 80 characters wide, you will see only 80 characters of the data.

In the Search command, period lines are considered trailer lines if they follow asterisk or tab lines.

Note: *Do not use period lines for data lines if you want to use the report with the Data Wizard.*

Special Lines

Type any character other than a tab, period, or asterisk in column one to specify a column-formatted, edited line. Special lines have all the other characteristics of tab lines. You can use special lines for data lines if you will use the report only with Cool ICE Script. Do not use special lines for data lines if you want to use the report with the Data Wizard.

Section 10
Data Wizard

This section introduces the Data Wizard and the application components it builds. It discusses the following topics:

- About the Data Wizard
- Application components
- Types of application components
- Output from Data Wizard
- ICE Admin and Data Wizard
- Data Wizard and security profiles
- Data Wizard functional flow

About the Data Wizard

The Data Wizard is a tool that provides a simple, interactive way to build an application component. An application component is a self-contained entity that

- Gathers data from up to five data sources
- Manipulates the resulting data as a table
- Formats this data as a form, table, or chart
- Interacts with the underlying data

When an application component meets your requirements, you can save its definition. The Data Wizard builds a Cool ICE data service that corresponds to the component definition. The Data Wizard also gives you the option of restricting access to the service. By assigning a profile to a component, you can control who can access and execute it.

Note: *To access the Data Wizard, you must be a registered user and a member of the ICE-Develop profile.*

Application Components

An application component is a self-contained entity that describes a sequence of executable steps. A step is the smallest unit of work that an application component can perform. Examples of steps are

- Selecting a data source
- Specifying a calculation to be performed
- Sorting data

A developer builds an application component by interactively adding steps to the component until it performs the desired business logic. An application component must contain at least one step but no more than 50. The sequence of steps in the application component determines the order of execution. Input to a step is the output of the previous step.

When building a new application component or editing an existing one, developers can add new steps, edit existing steps, or delete steps from the component. This makes the application component easy to modify as business needs change.

To simplify the process of building an application component, the Data Wizard lets an application developer view the current result of a step. An application component does not need to contain all of its steps before it can be executed. If the component executes successfully, the Data Wizard displays the current table, chart, or form for the developer to examine. If the component does not execute successfully, the Data Wizard displays an error message. This helps the developer isolate and fix any errors in the application component.

The developer can edit, add, or delete steps until a component produces the desired result. When a component is complete, the developer saves the application component and the Data Wizard builds a component definition and Cool ICE data service. See "Output of Data Wizard" for a description of the component definition and Cool ICE data service.

Types of Application Components

The Data Wizard builds two types of application components—standard and transaction.

Standard Components

A standard application component queries a database, manipulates the data, and possibly defines a data format.

Transaction Components

A transaction component is a component that is designed to work with a single record in a database. It allows the user to insert, update, or delete a record. A transaction component has two parts:

- *Selection* is the first part of a transaction component. In SQL, it is equivalent to a Select clause and any other processing as needed. The purpose of this part is to obtain a single record to work with.
- *Action* is the second part of a transaction component. In SQL, it is equivalent to an Insert, Update, or Delete statement. The action part starts with a single record (from the selection part of the transaction component) and an action—insert a new record, update an existing record, or delete an existing record.

The Data Wizard automatically builds both parts of a transaction component.

Output from Data Wizard

The output of the Data Wizard is a component definition and a Cool ICE data service. The Data Wizard creates both the component definition and the Cool ICE data service at the same time.

Component Definition

A component definition specifies the data sources you want to access, that is, the databases, tables, and columns. It also specifies how you want the data to appear and any operations you want to perform on the data. This includes the order in which the rows and columns appear, whether you want to view the data as a chart, any calculations you want to perform on the data, and so on. When you save the component definition, you specify a component definition ID that identifies the definition.

A component definition

- Contains a description of each step the service performs
- Points to the script of the corresponding service

The component definition ID is used as a parameter on the ExecuteDataService method of the Cool ICE Object. This method executes the service corresponding to a component definition and retrieves the data set the services produces. That data set is then available to the ASP page as an ActiveX Data Object (ADO) recordset. You can use any of the properties and methods of the ADO to manipulate the recordset.

Cool ICE Data Service

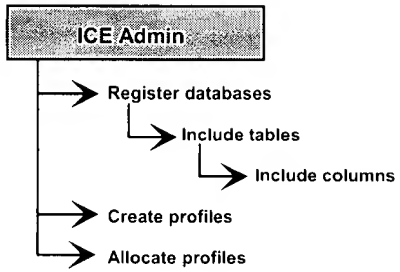
The Data Wizard builds a Cool ICE data service that contains the script for the component definition as a callable (@call) routine. The purpose of this service is to display in the user's browser the specified data whenever they execute the service. Each execution of the service retrieves the current data from the data sources and displays it in the user's browser. A service enables a user to monitor changes to the database over a period of time, for example, to monitor the results of an election.

ICE Admin and Data Wizard

ICE Admin performs the following tasks for the Data Wizard:

- Registers any databases you want to access
- Creates and maintains security profiles on databases, tables, columns users, services, and other objects in the Cool ICE repository

The following figure illustrates the tasks you complete with ICE Admin.



Data Source Registration

Before you can use the Data Wizard to access a local or remote database, you must first use ICE Admin to register the database connection. When you register a database connection, you enter the information Cool ICE needs to find and open the database, such as the name of the data source, its TCP/IP address, its type (ODBC, MQL, and so on), a user-id and password, and the name you will use to reference this data source in Cool ICE.

Using ICE Admin, you can also modify the registration details if the configuration of your system changes, for example, when you move the database to another server. You can also use ICE Admin to completely remove the connection to the database.

Once you register a data source, Cool ICE does not automatically grant access to the tables in a database and the columns in a table. As part of the data source registration, you must select which tables in the database and which columns in the tables you want to be able to access with the Data Wizard. Cool ICE enables you to individually select the tables and columns you want to be available to the Data Wizard.

Data Sources and Security Profiles

An administrator or developer, using ICE Admin, can associate security profiles with a registered database, tables within the registered database, or columns within a table. When a developer or administrator associates a profile with a database, table, or column, only those users whose user-ids are included in the profile can access that data source. A developer or administrator uses ICE Admin, the Cool ICE administration tool, to associate security profiles with data sources.

When a user selects a data source, the Data Wizard verifies that the user has the privileges needed to access that database and determines which tables and columns the user can access. If the user-id of the user is not included in any security profiles associated with the data source, the Data Wizard does not display that database, table, or column. The Data Wizard displays only those data sources for which a developer or administrator

- Did not associate a security profile, or
- Associated a security profile but included the user-id of the user in the security profile

See Also

See the *ICE Admin Help* for information on allocating profiles to a database, table, or column.

Data Wizard and Security Profiles

The security profiles that a user is a member of determine which services the user can execute. For the Data Wizard, there are two basic types of users: the developer who executes the Data Wizard and the end-user who executes the services created with the Data Wizard.

The Data Wizard can associate profiles only to component definitions and their corresponding services. Use ICE Admin to register users, create profiles, and associate those profiles with databases, tables, columns, services, and so on.

User Executing the Data Wizard

The following profiles affect the user who executes the Data Wizard:

- ICE-Develop

  Membership in this profile enables the user to execute the Data Wizard. The Data Wizard button on the Cool ICE main menu does not appear unless an administrator included the user-id of the person who wants to execute the Data Wizard in the ICE-Develop profile.

- Profiles allocated to data sources such as databases, tables, and columns

  If an administrator or developer associates a profile with a data source, Cool ICE grants access only to those persons whose user-ids are included in the profile. The developer who uses the Data Wizard can build component definitions only from the data sources they have privileges to access.

User Executing a Service Created by the Data Wizard

The following profiles affect the user who executes a service built with the Data Wizard:

- Profiles allocated to the service

  If a developer associates a profile with a service built with the Data Wizard, only those users whose user-ids are included in the profile can execute the service. The service appears on the Cool ICE main menu if the user's user-id is included in the profile.

- Profiles associated with data sources such as databases, tables, and columns

  If an administrator or developer associates a profile to a data source, Cool ICE grants access only to those persons whose user-ids are included in the profile. This means that if the developer of a service and the end-user of the service have different access privileges, they will see different results when they execute the service.

Gateway Configuration

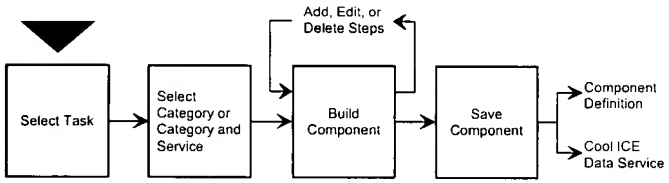
The configuration of the gateway determines the user-id of the person executing both the Data Wizard and the services built with the Data Wizard. You can configure the gateway to be either secure or unsecure:

- If the gateway is secure, Cool ICE displays a sign-on screen to solicit the user-id, department number, and password from the user. This user-id determines which Cool ICE services the user can access.

- If the gateway is not secure, Cool ICE uses the user-id that is configured as the default user-id. In the default configuration released with Cool ICE, the default user-id is WEBUSER. These users can execute all open services, that is, those services without a profile, and any services in which WEBUSER is a member of the profile.

Data Wizard Functional Flow

The following figure illustrates the functional flow used to build a component definition. The topics following the illustration describe the process used to build application components.



Start Data Wizard

The user starts the Data Wizard by clicking on the Data Wizard link on the Cool ICE home page. By default, Cool ICE displays the Data Wizard link only if the user's user-id is included in the ICE-Develop profile.

Select Task

The Data Wizard Home Page lists the tasks a developer can perform with the Data Wizard

Create a New Standard or Transaction Component

This task creates a new standard or transaction component when none of the existing components meets your requirements.

Create a New Component from an Existing Component

This task enables the developer to create a new standard or transaction component from an existing standard or transaction component.

Edit an Existing Component

This task loads an existing standard or transaction component so a developer can modify its steps, such as changing the data sources.

Execute a Component

This task retrieves the component definition, executes the service corresponding to it, and displays the data as a Cool ICE data service.

Change a Security Profile for a Component

After you save a component definition, you can use this task to change the existing security restrictions (remove profiles currently allocated to the component definition) or to further restrict the component definition (add additional profiles). The profiles control who can execute the service corresponding to the component definition. If you add a security profile to a component definition, a user cannot access that definition unless the profile contains his or her user-id. If a user does not have the privileges to execute a service,

- Cool ICE does not list the service in the categories and services on the Cool ICE main menu.
- Data Wizard does not list the service in its list of existing services that the user can execute or revise.

Delete a Component

This task permanently deletes from Cool ICE the component definition and Cool ICE data service built by the Data Wizard. If you used this component in another service, you must manually delete any references to it.

Select Category and Component

After the user selects a task, the Data Wizard displays the categories the user can access. For new components, this is the category where the developer wants to save the component. For existing components, this is the category where the component definition resides.

If a developer or administrator associated a profile with a category, the user-id of the person executing the Data Wizard must be included in that profile. Otherwise, the Data Wizard does not display that category.

If you are not creating a new component, the Data Wizard displays the component definitions in the category you selected. If a developer or administrator associated a profile with a component definition, the user-id of the person executing the Data Wizard must be included in that profile. Otherwise, the Data Wizard does not display the component definition.

Build Component—Select Sources

Create Variables

Using this step, a developer can specify any variables that are expected from the browser at the time the component is executed. The Create Variables step, if present, is always the first step of a component.

Select Source

Using the Data Wizard, you can build a component definition using any of the following data sources. You can access up to five data sources in a select source step. The Data Wizard verifies that you have the privileges needed to access the database, tables, and columns you selected. The Data Wizard checks for any security profiles allocated to the specified data sources and verifies that the user-id of the person executing the Data Wizard is included in the profile.

- SQL Relational Tables
  - ODBC (CORE level, 32-bit drivers)
  - Oracle
  - Sybase
  - Microsoft SQL
  - Informix
  - Ingres
  - Unisys MAPPER database
  - Unisys Relational Database Management System (RDMS)
  - Unisys A Series Query Language (ASQL)

  You must first register a database in ICE Admin before you can access it through the Data Wizard.
- Cool ICE Reports

  The Data Wizard can access only those reports that are registered through ICE Admin.

- Cool ICE Script

  A Cool ICE Script lets you create complex queries from one or more relational databases. See the *MRI Administration and User's Guide* for the commands you can include in the script and the *Cool ICE Developer's Reference* for information on using Cool ICE scripts with the Data Wizard.
- ActiveX Data Object (ADO) recordset

  A previous execution of an application component (or an ASP page) creates the ADO recordset. The recordset is saved using ICE Admin as a state management report. Using the Data Wizard, a developer can use the state management report as a data source.
- Last current table

  A developer can use the last current table (from an earlier select source step in the same application component) as a data source in a subsequent select source step.

When a developer selects data sources from the same database, the Data Wizard creates a single SQL select statement to request the combined view. This ensures that the database manager is used in the most efficient manner.

If a developer selects data sources from multiple databases, the Data Wizard fetches the data and stores it in the Cool ICE repository where it is joined using MAPPER Query Language (MQL).

Build Component—Manipulate Columns

Column Control

The Column Control step determines which columns appear in the table the application component builds. A developer can either add new columns to the table or hide existing columns.

Use Partial Columns

With the Use Partial Columns step, developers can select a subset of each column for use in subsequent operations.

Build Component—Manipulate Data

Perform Calculations

The Perform Calculations step lets developers compute, compare, and replace numeric data, character strings, dates, and times in selected columns.

Analyze Columns

The Analyze Columns step lets developers analyze and summarize selected data in the report or table. For each column you can compute a total value, average the data, select a minimum column value, or select a maximum column value.

This step is available only for standard components.

Sort Records

This step sorts the table or report according to the data in up to five columns.

Build Component—Display Data

Display a Record

This step displays a single record in a two-column format. The left column contains the field title and the right column displays the field contents.

If this step is part of a transaction component, the Insert Information or Update Information step determines whether a field is read-only or if a user can change the contents of the field.

Display Records

The Display Records step displays multiple records in a table. The first row of the table contains the field title. All other rows represent a record with each cell containing the content of the field.

If this step is part of a transaction component, the Insert Information or Update Information step determines whether a field is read-only or if a user can change the contents of the field.

Display Graph

The Display Graph step creates a chart of your data. The definition of the graph or chart is part of the component definition.

Build Component—Specify Transaction

Insert Information

This step is part of a transaction component only. When part of an application, it enables a user to insert a record into a database.

Delete Information

This step is part of a transaction component only. When part of an application, it enables a user to delete a record from a database.

Update Information

This step is part of a transaction component only. When part of an application, it enables a user to update a record in a database.

Save Component Definition

Once you are satisfied with the steps in the application component, the Data Wizard displays a screen that enables you to save the application component as both a Cool ICE data service and as a component definition.

From within the Data Wizard, you can execute the service, attach a security profile to the component definition, or return to the Cool ICE home page. If you exit the Data Wizard, you can reference the component definition in an ASP page.

Section 11
Graphics Server

The Cool ICE Graphics Server allows you to dynamically create business charts from a predefined set of data. The Graphics Server writes the chart to the image directory as a Graphics Interchange Format (GIF) file. A service can then display the chart in the browser.

This section discusses the following topics:

- Native Services and Graphics Server
- Open Services and Graphics Server
- Data Wizard and Graphics Server
- Chart Types

Native Services and Graphics Server

You can create charts directly from the current result (-0) of a Cool ICE native service. For native services, Cool ICE includes two graphing routines:

- Basic routine

  Provides the graphing capabilities delivered with previous versions of Cool ICE. Existing Cool ICE native services written to use the basic graphic routine continue to function as they did in the previous release of Cool ICE and produce the same charts. In addition, some new chart types are available with Cool ICE 2.1.

- Advanced routine

  Provides enhanced graphing capabilities, which allow you to specify individual properties of the chart you want to generate.

Refer to the *Cool ICE Developers Reference* for more information on creating charts from native services.

Open Services and Graphics Server

The enhanced graphing capabilities provided in the advanced graphing routine are also directly accessible from active server pages. An ASP page can access the graphing engine directly by creating an instance of the Cool ICE graphics server object and then using the properties and methods of the object to create a chart. Refer to the *Cool ICE Developers Reference* for more information on creating charts from open services.

Data Wizard and Graphics Server

You can use the Data Wizard to create charts as the output of an application component. The Data Wizard implements the capabilities of the basic graphics routine. Refer to the *Data Wizard Help* for information on how to create charts with the Data Wizard.

Chart Types

Using the Cool ICE Graphics Server, you can create the following types of charts. Each type has several variations.

- Area
- Bar
- Box-Whisker
- Bubble
- Candlestick
- Gantt
- High-Low-Close (HLC) and Open-High-Low-Close
- Line, Log/Lin, and Lin/log
- Pie
- Polar
- Scatter
- Surface
- Tape
- Combination (2D Bar and Line)

Area Charts

An area chart connects the data points of a data set with a line and then fills (shades or colors) the area bounded by the line. Area charts emphasize the relative proportions of the data sets. They show the value of each data set and the total value of all the data sets.

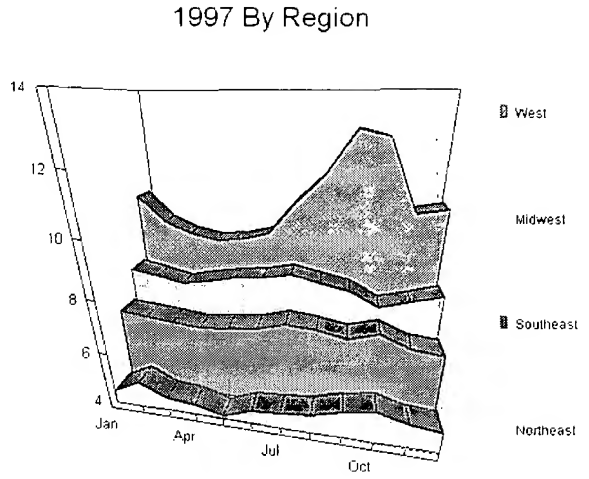
The X-axis is frequently a time progression, such as weeks, months, or years. Area charts are typically used to emphasize the magnitude of change over a period of time. For example, you could use an area chart to show how much each of the products in a product suite has contributed to the organization's profit and how the contributions have changed over the last three years.

Types of Area Charts

You can create the following types of two-dimensional (2D) and three-dimensional (3D) area charts:

- Stacked data sets (default)—plots the values for the Y axis for one data set relative to the values of the previous data set. That is, one data set appears stacked on top of another to show the relationship of each data set to the whole.
- Absolute (relative to the X axis)—plots the data points as their absolute value.
- Stacked percentage—plots the data set as a percent of the total
- Stacked semilogarithmic (2D only)
- Absolute semilogarithmic (2D only)

Illustration—Three-Dimensional Area Chart



Bar Charts

In a bar chart, rectangular bars represent a data category and its value. In a horizontal bar chart, the data categories are along the vertical (Y axis) and the values along the horizontal (X axis). Horizontal bar charts are used to compare the value of distinct data categories with each other. For example, a horizontal bar chart could show the number of products sold in each of the enterprise's geographic regions.

In vertical bar charts, the data categories are along the horizontal (X axis) and the values along the vertical (Y axis). Vertical bar charts are also called column charts. They are typically used to compare a single data category at different time intervals. For example, a vertical bar chart could show how many employees were in the organization at the end of each of the previous ten years.

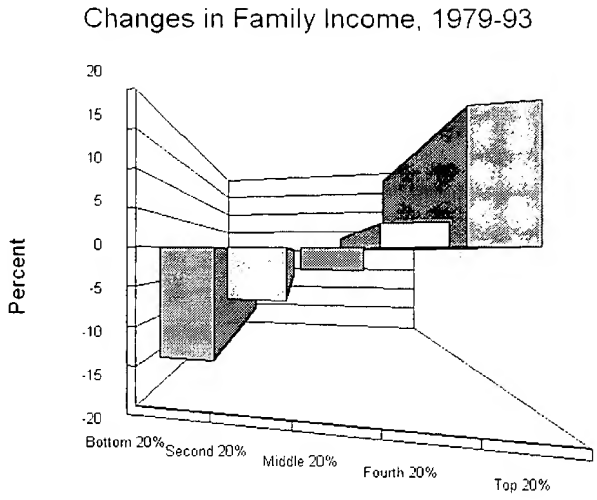
In a stacked bar chart, the rectangular bar contains more than one data category. Stacked bar charts show the relationship of the individual categories to the whole, and the change in both the value of the individual categories and in the value of the whole over the X axis.

Types of Bar Charts

You can create the following types of two- or three-dimensional bar charts:

- Vertical or horizontal bars
- Stacked vertical or horizontal bars
- Stacked % vertical or horizontal bars
- Vertical or horizontal stacked floating
- Vertical or horizontal Pareto
- Vertical or horizontal Z-clustered (three-dimensional only)

Illustration—Three-Dimensional Bar Chart



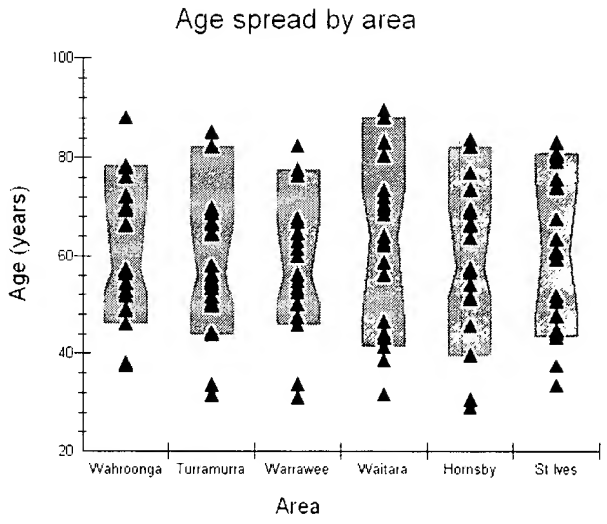
Box-Whisker Charts

Box-Whisker charts show the spread of data values around a median for one or more data groups.

Types of Box-Whisker Charts

- Raw data, samples drawn (default)
- Raw data, no samples drawn
- Parametric data
- No notch, whisker, or median line
- Black border

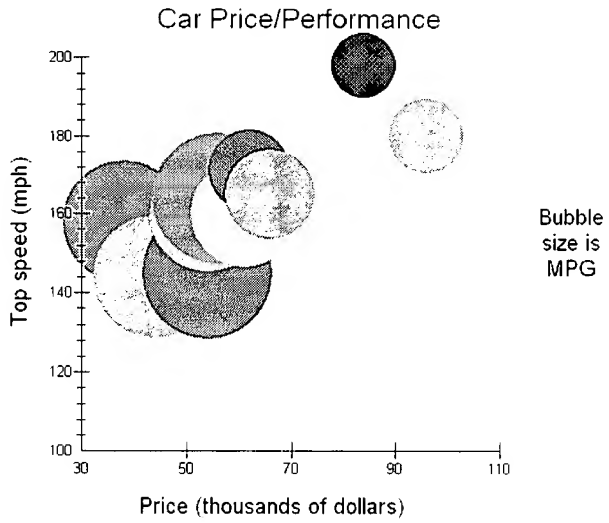
Illustration—Box-Whisker Chart



Bubble Charts

A bubble chart plots the values of a data set as a series of circles (the bubbles). A bubble chart is similar to a scatter chart, except the size of the bubble represents another dimension of the data. To create a bubble chart, you need three columns or rows of data: one to plot along the x axis, one to plot along the y axis, and one to represent the size of the bubble drawn at the x,y coordinate. Bubble charts are frequently used to represent financial data.

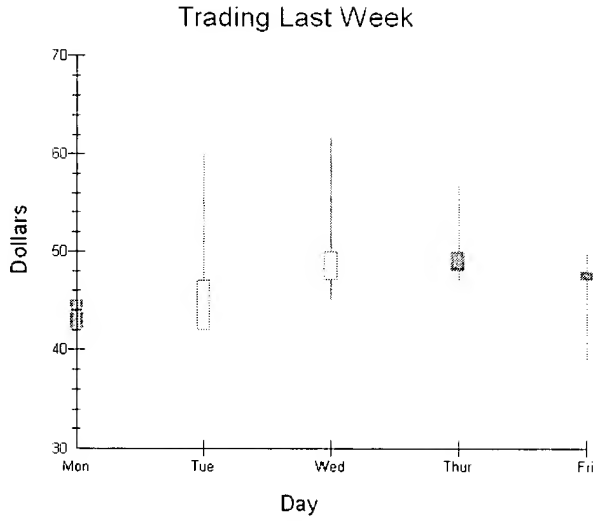
Illustration—Bubble Chart



Candlestick Charts

A candlestick chart uses boxes with lines extending from the top and bottom of each box to represent opening and closing values

Illustration—Candlestick Charts



Gantt Charts

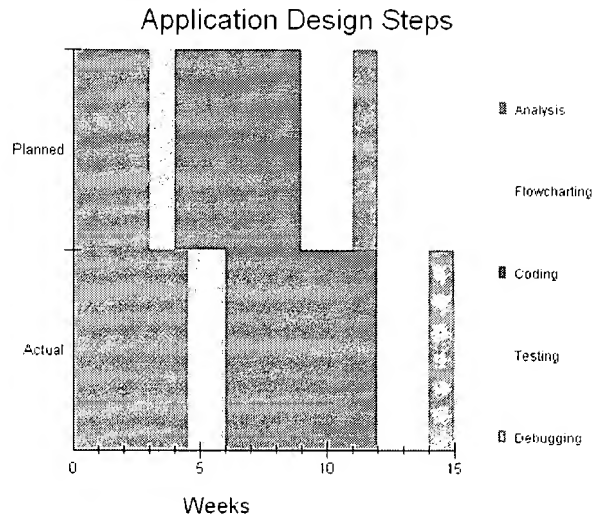
A Gantt chart is a horizontal bar chart that plots the tasks of a project against time. A bar represents the start and stop time for each task and its duration. The relative position of the bars indicate the sequence of the tasks. Gantt charts are typically used for project planning.

Types of Gantt Charts

You can create the following types of Gantt charts:

- Adjacent bars (default)
- Spaced bars

Illustration—Gantt Charts



High-Low-Close (HLC) and Open-HLC Charts

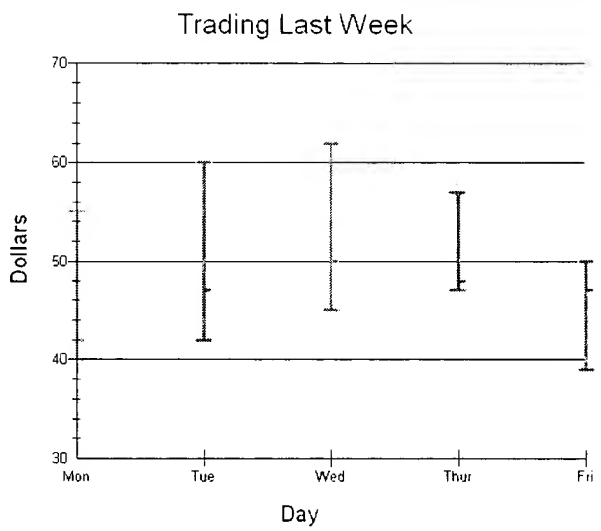
A high-low-close (HLC) chart plots the price of a commodity against time. The HLC chart represents the price range with a vertical bar. Small horizontal bars represent the high, low, and close. HLC charts are typically used to plot the price of a stock.

Types of HLC Charts

You can create the following types of HLC charts:

- Open (if used), high, low, and close bars (default)
- No open or close bars
- No high or low bars
- No bars at all

Illustration—HLC Charts



Line, Log/Lin, and Lin/Log Charts

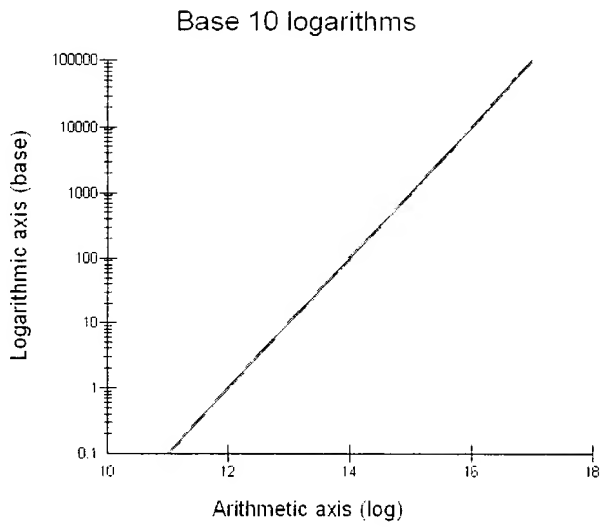
A line chart plots the data points of a data set and then connects the points to create a line. Line charts emphasize a trend in the data set. If the line chart contains multiple data sets, a line chart emphasizes the relationship among the different data sets. Line charts are frequently used to show how a variable changes over time or to compare a number of data sets.

Types of Line Charts

You can create the following types of line charts:

- Lines (default)—Connects the data points with a line
- Symbols—Plots the data points using a symbol
- Sticks—Plots the data points with a vertical line
- Sticks and symbols—Plots the data points with a vertical line and a symbol
- Lines and symbols—Plots the data points with a symbols and connects the symbols with a line
- Lines and sticks—Plots the data points with a vertical line and connects the vertical lines with a horizontal line
- Lines, sticks, and symbols—Plots the data points with a symbol and vertical line and connects the symbols with a horizontal line

Illustration—Log/Lin Chart



Pie Chart

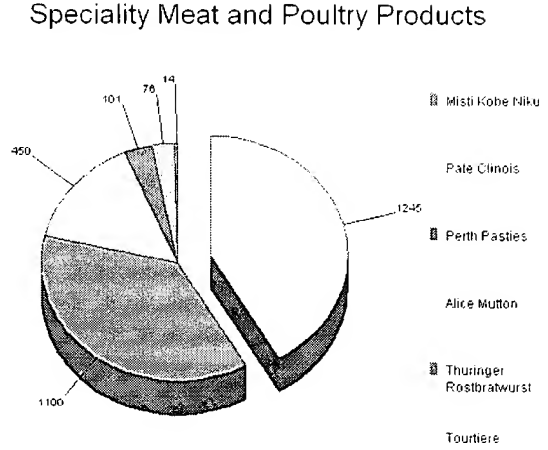
A pie chart plots one data set to show the relation of the individual data items to the entire data set. The individual data items are represented by wedges of a circle (slices of the pie). Pie charts are typically used to show the proportion of the individual data items in a data set to the whole data set at one specific moment in time. Each slice of the pie chart can also be expressed as a percentage. Use an area chart if you want to show the relationship of an individual data item to the entire data set over a period of time.

Types of Pie Charts

You can create the following types of two- and three-dimensional pie charts:

- Lines join labels (default)
- No label lines
- Colored label lines
- Percent (%) labels
- Percent (%) labels without lines
- Percent (%) colored labels
- Percent (%) colored labels without lines

Illustration—Three-Dimensional Pie Chart



Polar Chart

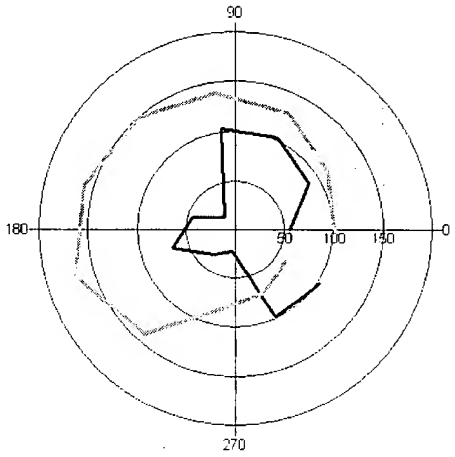
A polar chart plots the data points in a data set relative to a center point and to the other data points. They are like a circular line chart. Polar charts are also called radar or star charts. They are typically used to plot cyclical data where the X axis is a single continuous variable.

Types of Polar Charts

You can create the following types of polar charts:

- Default lines
- Symbols
- Sticks between points and origin (center)
- Sticks and symbols
- Lines
- Lines and symbols
- Lines and sticks
- Lines, sticks, and symbols

Illustration—Polar Chart



Scatter Charts

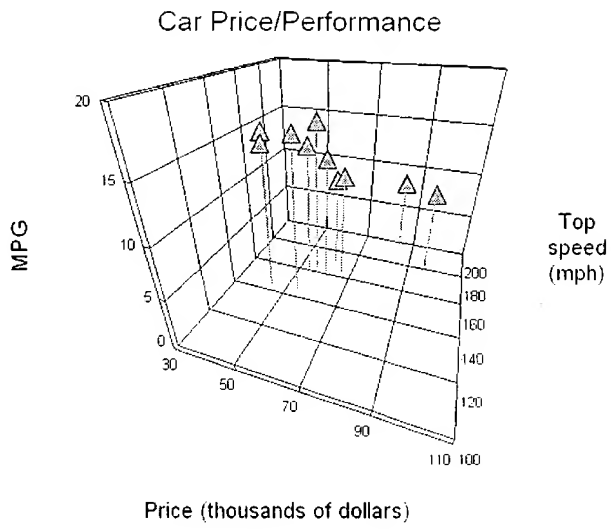
A scatter chart plots the data points in a data set against their values. A two-dimensional marker represents each data point. The data points are not connected with a line, bar, or any other type of graphic symbol. Data points from the same data set are shown with the same marker. Scatter charts are the clearest representation of a data set.

Types of Scatter Charts

You can create the following types of two- and three-dimensional scatter charts:

- Symbols only (default)
- Curve only (2D)
- Curve and symbols (2D)
- Symbols and vertical sticks (3D)
- Symbols and lines between points (3D)
- Symbols, vertical sticks, and lines between points (3D)

Illustration—Three-Dimensional Scatter Chart



Surface Charts

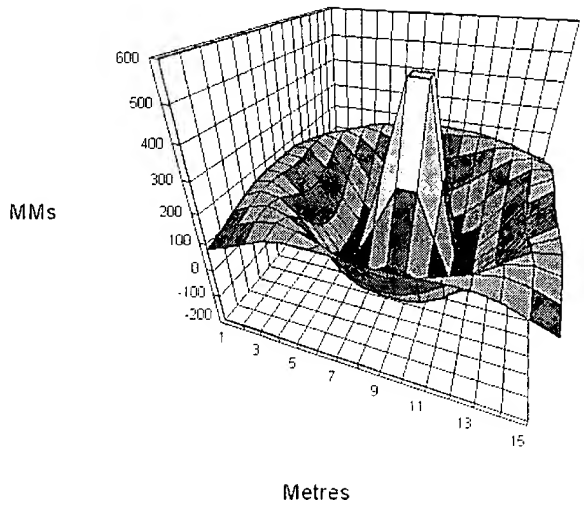
A surface chart plots three-dimensional data sets. The X and Y values determine the coordinates for the Z value. Color panels indicate areas that are in the same range of values.

Types of Surface Charts

You can create the following types of surface charts:

- Filled panels and boundary lines (default)
- Boundary lines only
- Filled panels only
- Filled panels and boundary lines with side wall
- Boundary lines only with side wall
- Filled panels only with side wall

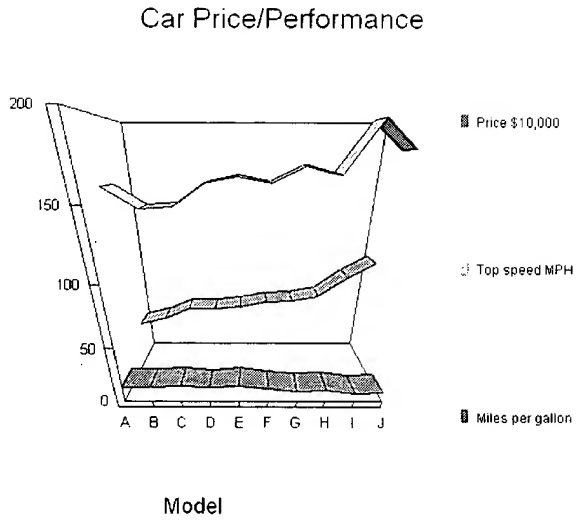
Illustration—Surface Chart



Tape Charts

A tape chart plots the data points of a data set and then connects the points with a rectangle. Like line charts, tape charts emphasize a trend in the data set. If the tape chart contains multiple data sets, it emphasizes the relationship among the different data sets. Tape charts are frequently used to show how a variable changes over time or to compare a number of data sets.

Illustration—Tape Chart



Section 12
Cool ICE Client

This section describes the Cool ICE Client. It discusses the following topics:

- Cool ICE Client
- Cool ICE Client scripts
- Multiple scripts
- ICE Admin and the Cool ICE Client
- Repository and Cool ICE Client

Cool ICE Client

The Cool ICE Client is a user interface program for the Windows 95 or Windows NT operating environments. It enables you to

- Access ICE Admin.
- View reports, tables, and other objects in the Cool ICE repository.

The Cool ICE Client supports access to either local or remote servers and the Cool ICE Engines running on those servers.

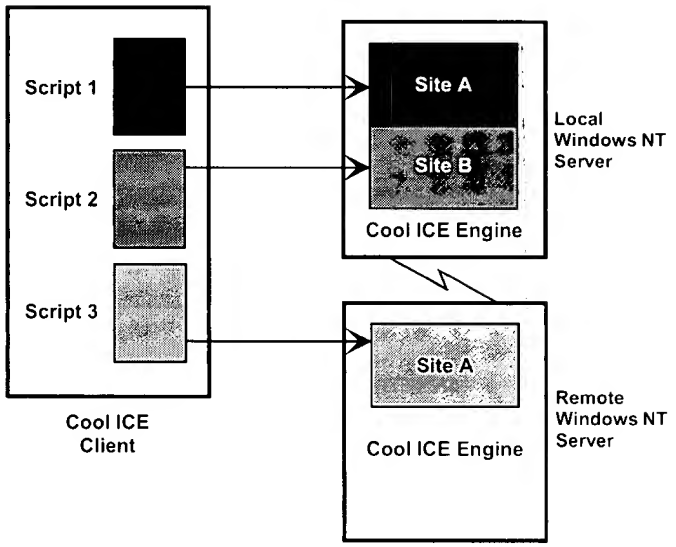
Note: *The Cool ICE Client is equivalent to the MAPPER Presentation Client that is released with MAPPER software.*

Cool ICE Client Scripts

The Cool ICE Client uses scripts to connect to the Windows NT server and the Cool ICE Engine. Before you can use ICE Admin, you must create a script. The script specifies the host, port, Windows NT login, password, site letter, and other information needed to establish a connection to the server and to the Cool ICE Engine running on that server. The Cool ICE Client includes a template to help you create the script.

Multiple Scripts

You can create multiple scripts. Each script could connect either to a different site on the same Windows NT server or to different Cool ICE Engines running on other Windows NT servers. You can create as many scripts as you need, but you cannot start more than 10 concurrent sessions from a single workstation. The following figure illustrates three scripts that connect to different sites and servers.



See Also

See the *Cool ICE Client Help* for information on creating a script and using the Cool ICE Client.

ICE Admin and the Cool ICE Client

You can either automatically start ICE Admin when your Cool ICE Client logon script executes, or you can manually start ICE Admin anytime after the logon script executes.

To automatically start ICE Admin, enter iceadmin in the MAPPER Run field of the script. The next time you execute the script, ICE Admin automatically starts.

To manually start ICE Admin, use your Cool ICE Client logon script to connect to a Windows NT server and a Cool ICE Engine. Then, select ICEADMIN from the Runs menu.

Repository and Cool ICE Client

You can use the Cool ICE Client to view the reports residing in the Cool ICE repository. The topic "Data Storage in Repository" explains more about the Cool ICE Client and the structure of these reports.

Section 13
Data and Transaction Access

Electronic business applications draw data from multiple back-end data sources. Many of these data sources are already accessible through standard technologies such as ODBC and SQL, or through DBMS-specific networking technologies. Developers could use available programming tools to build an application that accesses one or two back-end data sources whose data elements have simple, direct relationships. In a typical heterogeneous environment, the application might need to draw data from several sources, using several different technologies, and work through some complex indirect relationships to correlate the separate data items into an integrated view. Available programming tools do little to assist in this, forcing developers to write a lot of code to integrate the data.

This section describes the capabilities of Cool ICE software related to data and transaction access. It discusses the following topics:

- Data integration with Cool ICE
- Data sources
- Data access methods
- Data source configuration
- Data source security
- Database access connection pooling
- Transaction access

Data Integration with Cool ICE

Cool ICE assists in this data integration task. The Cool ICE Engine provides a place to hold and process data sets drawn from the various back-end data sources. The engine makes it easy for the developer to filter, manipulate, and correlate the source data sets to produce the desired result set. The Cool ICE Engine can access many popular data sources, including the leading open platform and mainframe database management systems. Application developers can use environment-specific technologies to access other data sources and move the data sets into the Cool ICE Engine for processing.

Data Sources

Cool ICE supports the following databases:

- ODBC (CORE level, 32-bit drivers)
- Oracle
- Sybase
- Microsoft SQL
- Informix
- Ingres
- Unisys MAPPER database
- Unisys Relational Database Management System (RDMS)
- Unisys A Series Query Language (ASQL)

Remote MAPPER databases use MRI to MRI networking.

Data Source Registration

To use a data source in Cool ICE, you must register it using ICE Admin. For the purpose of data source registration, Cool ICE considers a data source local if the

- Data source resides on the Web server running Cool ICE.
- Data source interface driver (ODBC driver) resides on the Web server running Cool ICE even if the physical database is on a remote server.

Cool ICE and the Relational Interface

The Relational Interface (MRI) is a feature of the Cool ICE Engine that gives users access to external data managers such as ODBC, Oracle, Informix, Sybase, Ingres, Microsoft SQL, and others. MRI lets you do the following:

- Retrieve data from an external data manager and bring it into the Cool ICE environment (Data Wizard and MRI statements).
- Insert, update, and delete data from tables stored in external data managers, without having to know anything about how the data is actually stored (MRI statements).
- Create or delete complete tables in external data managers (MRI statements).

Together, Cool ICE and MRI provide a seamless connection to external data. Using MRI you can access local databases, remote databases using networking between two MRI systems, and remote databases using the database vendor's networking capabilities.

Data Access Methods

Cool ICE contains two methods to access data—Data Wizard and MRI statements.

Data Wizard

The Data Wizard is a tool that provides a simple, interactive way to build database application components. Using the Data Wizard, you can build an application component by selecting one or more data sources, selecting specific columns to include, manipulating the data, and viewing or creating a graphic representation of the data.

The Data Wizard creates a component definition and a Cool ICE data service. The purpose of the component definition is to create a data set. You can use the component definition in an open service (ASP page) to retrieve a data set so that it is available to the ASP page as an ActiveX Data Object (ADO) recordset. The purpose of the Cool ICE data service is to display the specified data set whenever a user executes the service.

A data source for the Data Wizard can be either an SQL relational table, a Cool ICE report, a Cool ICE Script, ADO recordset, or last current table. (See "Build Component—Select Sources" in Section 10 for more information on Data Wizard data sources.)

MRI Statements

In Cool ICE native services, developers can use MRI statements to perform the full range of relational operations. These statements instruct Cool ICE scripts to open sessions with external data managers and to retrieve, create, or modify data. The following statements are available:

- Log On to Relational Database (LGN)
- Data Definition Information (DDI)
- Relational Aggregate Fetch (FCH)
- Relational Aggregate Modify (RAM)
- Submit SQL (SQL)
- Trace Relational Syntax (TRC)
- Log Off Relational Database (LGF)

See the *MRI Administration and User's Guide* and the *Command Reference* for more information on these statements.

Comparison of Data Access Methods

The following table compares the two methods of data access. The method you choose depends on the experience of your developers, the type of service you want to build, and what operations you want to perform on the data.

	Data Wizard	MRI Statements
Experience Level	Novice	Advanced
Service Type	Native and Open	Native
Relational Operators	Retrieve and view data; insert, update, and delete single records	Full (retrieve, view, and update data)

Data Source Configuration

The following steps outline the basic process of configuring a data source for use in Cool ICE.

1. Set up the data source.

   Using the instructions provided by your database vendor, set up the data source. For existing data sources, the Cool ICE application developer needs to know the user-id, password, and possibly other information depending on the data source.

2. Configure the network.

   If the database is on a remote server, configure the network between the Web server running Cool ICE and the remote host.

3. Configure and register the ODBC source name.

   If you are using ODBC to connect to a data source, configure and register the ODBC source name. For an ODBC data source that resides on a remote host, the developer must register the data source name both in the ODBC applet on the Windows NT Web server and in MRIDBA (A script that performs administration and configuration tasks for MRI). The data source names must match. See "Configuring and Registering an ODBC Source Name" in the *MRI Administration and User's Guide* for more information.

4. Verify that you can access the data.

   Make sure that you can access the data through some method other than Cool ICE. This ensures that the data source and network are correctly configured.

5. Register the database in ICE Admin.

   See the *ICE Admin Help* for information on registering the data source with Cool ICE.

6. (Optional) Allocate security profiles.

   Allocate security profile to the entire database, or specific tables or columns. See the *ICE Admin Help* for information on allocating a profile to a data source.

Data Source Security

Cool ICE also provides security for accesses to back-end data sources. Each data source has its own unique logon or access control requirements based on the administrative policies for that back-end system.

The Cool ICE security profile determines which back-end data sources each user or class or user can access. A developer can use profiles to control access to an entire data sources, specific tables within the database, or columns within a table. There is not relation between the Cool ICE user-id and the user-id and password used to access the data source. The Cool ICE user-ids and security profiles determine which services a user can execute. The service or component definition must contain the user-id and password needed to access the data.

See the *ICE Admin Help* for information on including a data source in a profile.

Database Access Connection Pooling

One component of the Relational Interface (MRI) is a low-level server process called the Relational Interface Manager (MRIM). MRIM provides the connection between Cool ICE services and the relational database management systems supported by MRI.

Cool ICE contains an optional and advanced feature called database access connection pooling. It is implemented as a pool of prestarted MRIM processes. This feature improves the performance of Cool ICE services that access external data sources. When a script or service connects to a database, the system selects a prestarted MRIM process from the pool. This avoids the costs associated with starting a standard (not prestarted) MRIM process. If a prestarted MRIM process is not available, the system starts a standard MRIM process.

The prestarted MRIM process pool is available only for Cool ICE sites on the local machine. Requests from remote Cool ICE sites continue to use MRI to MRI networking (that is, mapethd starts a standard MRIM process).

You can configure MRIM so all Cool ICE requests go through a pool of prestarted MRIM processes and wait if a prestarted process is not available. This may be best in a heavily used system because starting a standard MRIM process when a prestarted MRIM process is not available uses system resources. Since waiting for a prestarted process to become available does not increase the load on the system, the system can accomplish more work.

You configure database access connection pooling using MapAdmin. See the *MapAdmin Help* for information on how to configure a pooled of MRIM processes.

Accessing ODBC Data Sources through the MRI Process Pool

If you access an ODBC database through the prestarted MRIM process pool, you can configure MRIM to keep a specified number of ODBC connections open for reuse by subsequent Cool ICE services. By using a pool of prestarted MRIM processes, you avoid the database connection costs for subsequent references to that ODBC data source.

To access ODBC data sources through the prestarted MRIM process pool, you must

- Specify values for the **Maximum ODBC connections to keep open** and **Maximum ODBC disconnects before MRIM restarts** parameters in the local parameter configuration file. You use MRIDBA, another component of MRI, to set these values. See "Setting Up a Local Parameter Configuration File" in the *MRI Administration and User's Guide* for more information on these parameters.
- Configure and start a pool of MRIM processes. See the *MapAdmin Help* for information on configuring the number of prestarted processes in the MRIM process pool and on starting the configured MRIM processes.

Transaction Access

Cool ICE supports the following implementations of online transaction processing.

Open/OLTP

Open/OLTP software is a standards-based suite of middleware products that implement mission-critical distributed transactions processing applications in a client-server environment. Using Open/OLTP, an enterprise can develop, deploy, and administer transaction processing applications. Open/OLTP products are supported on the following Unisys enterprise server platforms:

- ClearPath IX servers
- Standalone 2200 systems

Open/OLTP is based on the X/Open Distributed Transaction Processing (DTP) model that is also supported in products from other vendors. The most prominent of these is BEA TUXEDO, which is supported on Windows NT, UNIX, and a few proprietary operating systems.

BEA Tuxedo

BEA TUXEDO is an application development platform that enables developers to create applications that span multiple hardware platforms. It provides a scaleable, proven enterprise messaging and transactional framework to handle mission-critical applications.

Transaction Access from Cool ICE Services

Cool ICE enables application developers to access OLTP services from either

- Open services—Using a COM object in an active server page.
- Native services—Using the commands in Cool ICE Script.

The application developer can also allocate security profiles to the services. The security profile determines who can execute the services.

Transaction Access from Open Services

Accessing a transaction service from a Cool ICE open service involves the following steps:

1. Install DGate, DGateAce, and the connectors.

   See the *Cool ICE Getting Started* for information on installing these components.

2. Create the DCOM server.

   Create the DCOM server using DGateAce and the Microsoft Visual C++ ATL COM AppWizard.

3. Create the active server page.

   Create the active server page using VBScript and HTML. The ASP page solicits information from the user and displays the resulting data.

OLTP Example

For an example of creating a DCOM server and active server page, see the OLTP Employee Demo installed with Cool ICE. To start the example, click the Cool ICE Examples category on the Cool ICE home page and then OLTP Employee Demo. The example shows how to access an employee database from the Web using active server pages and ActiveX technology. Cool ICE supplies the following files to help you access transactions from the Web:

- "Web Access to OS 2200 Transactions Using Active Server Pages" describes how to create the DCOM server and active server page in more detail. To view this document, start the OLTP Employee Demo, click System Diagram, then click Development Process.
- Employee.asp is the active server page used by this example. The default folder for this file is \cool-ice\default gateway\Examples\OLTP on the server where you installed Cool ICE.

Key Components

Cool ICE uses a gateway and connectors to access Open/OLTP services and BEA TUXEDO services from Web pages that use the Microsoft distributed component architecture (DCOM).

DCOM/OLE Gateway (DGate)

Gateways are envoys between the outside world and various resources on servers. Generally, gateways convert input data from Web pages into VIEWs that are understood by Open/OLTP software and BEA TUXEDO.

Cool ICE includes the DCOM/OLE gateway (DGATE). DGate is the envoy between Distributed Component Object Mode (DCOM) clients and Open/OLTP services. DGate makes it possible to access the following types of host applications from a DCOM environment:

- Open/OLTP applications (local)
- Open/OLTP applications that participate in distributed transactions with other platforms running Open/OLTP and BEA TUXEDO software.

Connectors

Connectors are program extensions that take input data from gateways, package the data into standard Open/OLTP service requests, and send requests to Open/OLTP software through one of the following communications packages:

- High-Performance Transaction Processing Interconnect—The HTP/ic connector enables applications in the Windows NT environment to execute transactions against OS 2200 applications that use the Open/OLTP Transaction Manager. HTP/ic is based on XATMI and OSI TP standards.
- BEA TUXEDO Workstation Connector—The workstation connector (/ws) provides access to BEA TUXEDO services from clients resident on a Windows NT platform.

DCOM Client and Server

The DCOM client program provides a user interface for submitting transaction requests to the DCOM server and viewing the data it returns. In Cool ICE, the DCOM client program is built as an active server page.

The DCOM server program, at a minimum, accepts requests from DCOM clients, repackages the parameters into the format required by Open/OLTP transactions, and then forwards them over a named pipe to DGate.

DGate includes a utility called DGateAce that simplifies the process of defining the DCOM server programs.

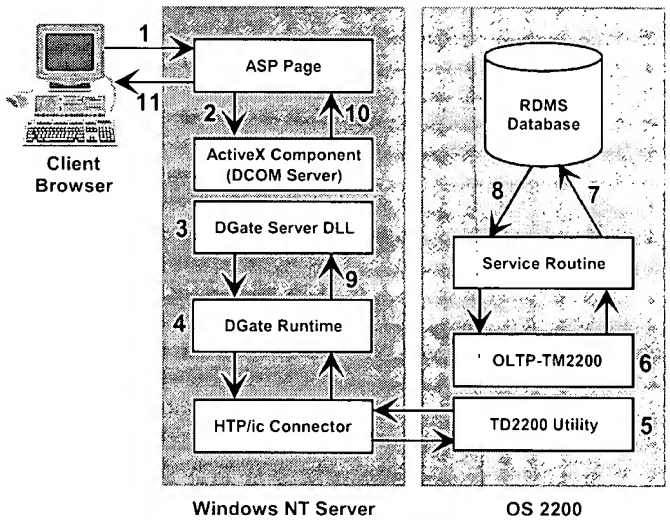
DGateAce Utility

DGateAce is a transaction gateway utility that assists Unisys customers who use standards-compliant client/server tools. DGateAce provides an interactive interface that simplifies building functions and the corresponding interface definition library (IDL) file to access client/server transactions in the OS 2200 environment. The DGateAce utility produces the following output files:

- Interface Definition Language (IDL) file—For Cool ICE, this file is not used. Instead, use the IDL file generated by the Microsoft C++ Active Template Library (ATL) COM AppWizard.
- Function files—For Cool ICE, the application developer cuts the function code from this file and pastes it into the implementation code section of the ActiveX component generated by the Microsoft C++ ATL COM AppWizard.

Illustration—Runtime Description

The following figure illustrates the runtime flow to access a transaction in an OS 2200 environment. A description of the numbered steps follows the illustration.



1. The client browses to the ASP page. The ASP page contains HTML and server-side scripting, such as VBScript.
2. The CreateObject method is issued to create and references the ActiveX component (DCOM server). The DCOM server provides access to the objects and methods available to the active server. The CreateObject method of the DCOM server creates an instance of the DCOM component on the server. The DCOM server was built with implementation code created by DGateAce.

3. The ActiveX component (DCOM server) calls the DGate server DLL (DGATE_server.dll) to transfer OLTP requests from the DCOM server to the DGATE runtime (DGate.exe). DGate server DLL provides access to functions that transfer Open/OLTP requests from the DCOM server to the DGate runtime (DGate.exe). The DGate server DLL is included with DGate.
4. DGate runtime (DGate.exe) fields the request and transfers them through the HTP/ic connector to the 2200. DGate runtime acts as a conduit between the DCOM environment and the Open/OLTP environment. The DGate runtime is included with DGate.
5. TD2200 utility accepts the service request and passes it to OLTP-TM2200.
6. OLTP-TM2200 calls the service routine.
7. The service routine accesses the 2200 RDMS database and performs its service.
8. The service sends a response back to the DGate runtime which reformats the response into an output structure and returns it to the DGate server DLL.
9. The DGate server DLL releases its connection to the DGate runtime and returns the output to the DCOM server.
10. The DCOM server function call completes and returns output to the *.asp file.
11. Microsoft Internet Information Server (IIS) formats the HTML and sends it back to the client browser.

Transaction Access from Native Services

By using the Cool ICE scripting language in native services, an application developer can combine the capabilities of the Cool ICE Engine—such as reporting, database access, and information analysis—with Open/OLTP distributed transaction processing. Cool ICE Script supports the Open/OLTP environment by providing the following features:

- @TPC statement (client interface)

  A statement that implements a subset of the TUXEDO ATMI interface (functions that are useful for developing Open/OLTP client programs).

- @TPS statement (server interface)

  A statement that implements a subset of the TUXEDO ATMI interface (functions that are useful for developing Open/OLTP server programs). This feature is available on a custom install and only if you select OLTP files, then OLTP Client/Server. See the *Cool ICE Getting Started* for more information.

- Typed buffers

  Cool ICE Script currently supports X_OCTET, FML, STRING, and CARRAY typed buffers. Cool ICE Script does not support VIEW buffer types.

  Note: *Because the Cool ICE repository does not have the ability to perform a two-phase commit, it cannot participate in a global transaction where a rollback of all or part of a transaction is required.*

See Also

Refer to the *Command Reference* for more information on the @TPC and @TPS statements.

Section 14
Development Process

This section describes the Cool ICE development process. It discusses the following topics:

- Front-end application development tools
- Back-end application development tools
- Development process

Front-End Application Development Tools

An application developer can call the Cool ICE environment from a wide variety of application development tools, including

- Web authoring and scripting tools such as Microsoft FrontPage or Macromedia Dreamweaver
- Visual development and scripting tools such as Borland IntraBuilder
- Wizard-driven generators such as NetDynamics
- Java and VBScript scripting languages

All front-end application development tools access the Cool ICE Engine through COM-based interfaces. This interface enables new tools and new versions of tools to be added to the Cool ICE environment at any time. The application developer can also use the latest Web technologies and features such as

- HTML and vendor-specific HTML extensions
- Java, ActiveX, and CORBA
- Secure Sockets Layer (SSL), Secure HTTP, and digital certificates signatures
- State-oriented connections
- Push

Back-End Application Development Tools

Developers can call the Cool ICE environment from a wide variety of back-end application development tools, including

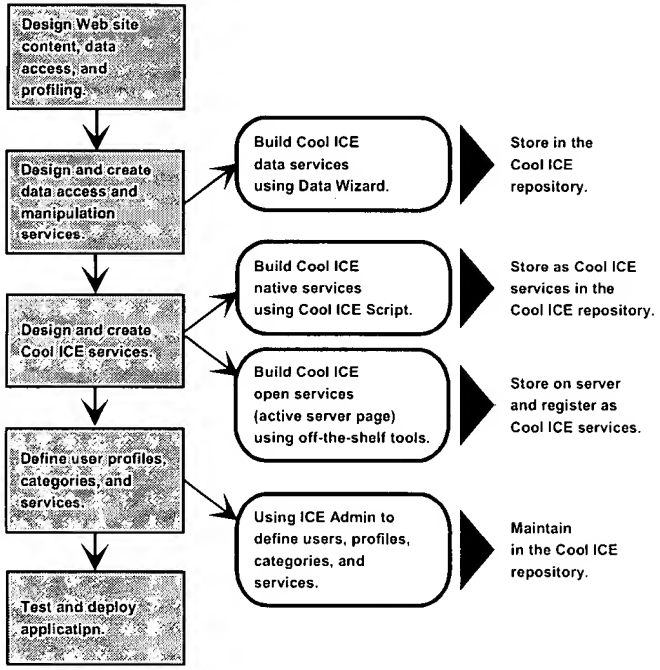
- Visual Basic
- COBOL
- Java
- C++
- Java scripting languages
- VBScript scripting languages

Development Process

This section describes the stages of a development process application developers can use to create a applications with Cool ICE. It focuses on using the features and capabilities of Cool ICE to implement the enterprise's business objectives for the site.

Illustration—Development Process

The following figure illustrates the five stages of the Cool ICE development process.



Stage 1: Design Web Site Content, Data Access, and Profiling

The first stage of the development process focuses on the design and content of the entire Web application. In this stage, the developer completes the following tasks:

- Determine business objectives

  What business process or logic will this Web site implement? Do you want your site to sell products or services? generate sales leads? supply product or service information to customers and suppliers?

- Determine audience

  For whom are you developing this application? What users or groups of users is the target audience for this application? What data does each user or group of users need? Are there users who should not access your site or part of your site? In later stages of the development process, you will use your analysis of the audience to implement Cool ICE security profiles.

- Identify data sources

  Where does the data reside that your application needs? What type of database it is? Can you access through the network? Will you pull information from the data source and push it to the user, or will you perform some additional manipulation of the data?

- Establish overall site design and organization

  How will your application solicit and present data to the user? How will the user navigate the Web site? What is the structure or hierarchy of the HTML pages in the Web site? Your local bookstore or the Internet contains information to help the developer design the Web site and create the user interface.

Stage 2: Design and Create Data Access and Manipulation Services

Data access and manipulation is one of the most important aspects of the Web application. The second stage of the development process focuses on design and creating the data access and manipulation services. This stage and the following stage, creating the service, are typically performed in parallel. To design and create data access and manipulation services, the developer completes the following tasks:

- Analyze your data sources

  Determine what data your application needs and the databases, tables, and columns that data resides in.

- Define any additional data manipulation

  Determine if you need to further manipulate the data. This may include joining tables from different data sources, performing calculations on the data, and so on.

- Register data sources

  Using ICE Admin, register the data sources you will access. This includes registering any tables or columns.

- Build data services using Data Wizard

  Use the Data Wizard to build an component definition, a Cool ICE data service, or both.

Stage 3: Design and Create Cool ICE Services

In the third stage of the development process, the application developer designs and creates the Cool ICE native or open services. This involves the following tasks:

- Select the type of service

  The application developer needs to determine which type of service is better for the application—native or open. The following factors influence this decision:

  - Developer experience—Developers familiar with Cool ICE script may be more productive writing native services while developers familiar with VBScript or a Java scripting language may be more productive writing open services.
  - Capability—Do you need the ability to reference other COM objects? If so, open services may be more appropriate.
  - Performance—A native service, since it interfaces directly with the Cool ICE Engine, may provide better performance. Depending on the service, the performance gain may or may not be significant.
  - Tools—Native and open services require different set of tools that must be available to the developers.

- Create the services

  Write the services using the scripting language appropriate for the type of service you are developing. If appropriate, modify one of the service templates or examples stored in the Cool ICE repository. Include any component definitions that were built in the previous stage in the service.

- For open services, register a virtual directory and create an alias for it.

  If you are using active server pages, register any virtual directories on the Web server (see your Web server documentation). The virtual directories are used to store the ASP pages you create. The Cool ICE installation creates a default virtual directory that you can use without modification. If you create a virtual directory, use ICE Admin to define an alias for it. Refer to *ICE Admin Help* for the information on the ASP directory alias.

- Configure the gateway

  Gateways allow users access to virtual directories containing active server pages or categories containing Cool ICE native services. Each virtual directory you create requires a gateway. The Cool ICE installation creates a default gateway. Refer to the Gateway Configuration Tool Help for information on creating additional gateways.

Stage 4: Define User Profiles, Categories, and Services

In the third stage of the development process, the application developer creates the category for the application and specifies any security profiles. By default, the Cool ICE system is open; anyone with access to Cool ICE can execute a service. However, if you want to restrict access to an object (the entire system, a category, or a service), you can create and allocate a security profile. The security profile specifies which users can access that object.

- Create the category

  A Cool ICE category stores the related objects and services for the Web application. Managing an application is easier when you store all objects relating to the application in the same category. To create a category, see the *ICE Admin Help*.

- Add services and other objects to category

  Add the services and other objects related to the application, such as images and applets, to the category.

- Register users

  To implement security profiles, you need to register those users to whom you want to grant special access privileges. You do not need to register all users. Unregistered users can access all open services, categories, or systems. A service, category, or system is open if you do not allocate a security profile to it. To register a user, see the *ICE Admin Help*.

- Create profiles and add users

  Use ICE Admin to specify the name and description of a new profile. Add users to the profile. See the *ICE Admin Help* for information on creating the profiling and adding users.

- Allocate the profile

  Use ICE Admin to allocate the security profile to those objects to which you want to control access. See the *ICE Admin Help* for more details.

Stage 5: Test and Deploy Applications

The last stage of the development process is finding bugs, making revisions, and retesting the application before its official launch. In this stage, the application developer completes the following tasks:

- Test the application

  The new Web technologies and multiple browsers with different capabilities require that you spend more, not less, time testing the application. As you develop a test plan, consider the following:

  - Does your site accommodate different browsers? How about different monitor resolutions and sizes?
  - Do all hypertext links work?
  - Are profiles correctly allocated to the services?
  - Is performance acceptable?

- Debug services

  To help debug services, an application developer can use the error log Cool ICE maintains. Whenever a service fails because of a syntactical error in the service, Cool ICE records the error in the error log. The error log provides information about the service at the time it failed; for example, the date and time of failure, who requested the service, and internal values from the service.

  Application developers can also use the trace log to debug a Cool ICE native service. The trace log enables the developer to run a service in the foreground. See the *ICE Admin Help* and the *Cool ICE Developer's Reference* for more information on using the error and trace logs to debug scripts.

- Deploy the application

  Keep the following in mind as you deploy your site:

  - Make sure you have a method to distribute and promote the URL for your site.
  - Depending on the volume of hits your site receives, you may need to tune the performance of your Web server.

- Monitor the application

  Once you deploy your site, you can use the access log Cool ICE maintains to provide information about who is using your site. Cool ICE logs all requests for services made from a browser with information about who requested the service, at what time, and how long it took Cool ICE to execute the service. See the *ICE Admin Help* and the *Cool ICE Developer's Reference* for more information on using the access log.

Glossary

A

access log
In Cool ICE, the record of requests for services made from a browser. The access log contains information associated with each request, such as who requested the service, at what time, and so on. The Event Viewer menu in ICE Admin allows administrators and service developers to view the access log.

Active Server Pages (ASP)
(1) The server-side execution environment in Microsoft Internet Information Server that executes ActiveX Scripts and ActiveX components on a server. (2) Developers can combine HTML pages, script commands, and COM components to create Web-based applications.

ActiveX
A set of language-independent interoperability technologies that enable software components written in different languages to work together in networked environments. The core technology elements of ActiveX are COM and DCOM. These technologies are licensed to The Open Group standards organization, and are being implemented on multiple platforms.

ActiveX control
A compiled software component based on the Component Object Model (COM) that encapsulates a set of business or user interface functions. An ActiveX control is used to provide user interface components and is designed to run on the client computer. These were formerly called OLE Controls. Optimizations in the technology resulted in smaller, faster software components and support for key features used to distribute components "just in time" over networks such as the Internet. ActiveX controls can be embedded in Web pages for use over the Internet as well as combined to create client/server applications that run over a corporate network. They can be created by a variety of programming languages from Microsoft or from third-party vendors. ActiveX controls use the file extension .ocx.

ActiveX component
A compiled software component based on the Component Object Model (COM) that encapsulates a set of business functionality. The functionality in an ActiveX component is accessed through ActiveX Automation interfaces. The ActiveX component can execute either on a client computer or on a server computer, transparent to the calling application, through DCOM. ActiveX components can be driven by a scripting language such as VBScript or JScript. All Java applets, running in the Microsoft virtual machine for Java, are automatically ActiveX components and use the file extension .class. ActiveX components that run within the calling application process use the file extensions .dll or .ocx. ActiveX components that run outside of the calling application process use the file extension .exe. *See also* COM.

ActiveX Data Object (ADO)
A set of object-based data access interfaces optimized for Internet-based, data-centric applications. ADO is based on a published specification and ships with Microsoft Internet Information Server and Microsoft Visual InterDev.

ActiveX server component
An ActiveX component designed to run on the server side of a client/server application. *See* ActiveX component.

ADO
See ActiveX Data Object.

applet
(1) A small, single-function application designed to be executed from within another application. (2) An HTML-based program built with Java that a browser temporarily downloads to a user's hard disk, from which location it runs when the Web page is open.

application
In Cool ICE, a set of related tasks accomplished by one or more services and the associated database, for example, an order-entry system.

application component
A self-contained script that can fetch data from multiple data sources, manipulate the resulting data as a table, and format this data as a form, table, or graph. For data from a single source, a developer can create an application component that updates, deletes, or inserts a single record in the database.

After creating the application component, the developer can save the application component for use as a Cool ICE data service and as a component definition. An application component can be as simple as a Select Source step that defines the data source, or can include up to 50 steps. *See also* step, data service, and component definition.

ASP
See Active Server Pages.

B

BEA TUXEDO /Workstation (/WS)
Software that calls an Open/OLTP batch service routine in the OS 2200 environment to access an RDMS 2200 database.

browser
See Web browser.

C

cabinet
In the Cool ICE repository, a group of eight drawers (B through I) referred to by a number, for example, cabinet 200.

cascading style sheet (CSS)
A definition of the layout and appearance of the elements in a Web page, such as its typeface, line spacing, margins, and so on.

category
A container for one or more Cool ICE services. A category typically contains an application.

CGI
See common gateway interface.

client side
Refers to any operation that is performed at the client in a client/server relationship. *Contrast with* server side.

column
A vertical line of information or field in a relational table. A component of a relational table where like data is stored. *See also* field *and* table.

common gateway interface (CGI)
A server-side interface for initiating software services. A set of interfaces that describes how a Web server communicates with software on the same computer. Any software can be a CGI program if it handles input and output according to the CGI standard. *See also* gateway.

Component Builder
In the Data Wizard, refers to the central screen for creating and editing an application component. After gathering initial Web variables, the Component Builder produces a list of steps for the component and offers the developer an opportunity to change the component. Changes include such actions as add, edit, or delete steps from the component. *See also* Web variables.

component definition
A Cool ICE report that contains the defining parameters required to build any specific request for a table, view, or action. The Data Wizard uses this information to construct script code that produces a table. When a developer saves an application component, the Data Wizard saves the component definition. A developer can then use the name (ID) of the component definition to create an ADO recordset viewed or updated through an ASP. *See also* Component Builder, application component.

Component Object Model (COM)
The object-oriented programming model that defines how objects interact within a single application or between applications. In COM, client software accesses an object through a pointer to an interface—a related set of functions called methods—on the object. *See also* Distributed Component Object Model.

connection
A link between the browser and Web server that enables a browser to request and receive an HTML page from a Web server. The connection ends when the Web server returns the HTML page. *Contrast with* session.

cookie
A small data file sent to the browser by a Web server. The browser stores the file on the workstation's hard drive. It sends the file back to the Web server each time it requests a page from the server. The file can contain information about the user's previous selections at that Web site and any personal preferences. Using cookies, a server or script can maintain state or status information on the client workstation.

Cool ICE configuration
A set of parameters used in establishing a connection to the Cool ICE Engine. A Cool ICE configuration defines characteristics such as a session timeout value, a Cool ICE connection pool name, default user-id/password for anonymous access to Cool ICE services, and various other parameters.

Cool ICE connection
A link between the Cool ICE Object and the Cool ICE Engine. The connection ends when the Cool ICE Object is destroyed or the connection is explicitly released. Cool ICE connections are maintained in a Cool ICE connection pool.

Cool ICE connection pool
A pool of active Cool ICE connections that the Cool ICE Object uses to establish a link to the Cool ICE Engine.

Cool ICE Engine
The component of Cool ICE that provides analytical functions, and access to data and applications.

Cool ICE gateway name
A name used by the Cool ICE Object to determine which Cool ICE configuration establishes the connection to the Cool ICE Engine. Several Cool ICE gateway names can be associated with a Cool ICE configuration. The Cool ICE gateway name is specified as part of the URL used to indicate which Cool ICE service to execute.

Cool ICE graphing engine
The component of Cool ICE that creates a graph from information stored in the current result. The Cool ICE service displays the graph in the browser.

Cool ICE Object
The component of Cool ICE that provides access to the Cool ICE Engine, Cool ICE native and data services, state management, and profiling.

Cool ICE repository
In Cool ICE, the database used to store services and other objects. The database is organized into a series of cabinets, drawers, and reports.

Cool ICE Script
The native, interpreted scripting language for writing native Cool ICE services. Data services created by the Data Wizard also use Cool ICE Script. The Cool ICE Engine executes the commands in Cool ICE Script.

Cool ICE Script Editor
A Windows-based editor for Cool ICE Script, the native scripting language. A service developer can configure ICE Admin to start the Script Editor whenever he or she selects Modify Service on the Object Attributes dialog box.

Cool ICE scripting language
See Cool ICE Script.

Cool ICE Service
An object in the repository that contains any combination of HTML, client-side scripting, and server-side scripting. Cool ICE executes the service when requested by a browser. A service performs some business logic and returns a result. Cool ICE contains three types of services: native, open, and data.

Cool ICE Service Handler
The Cool ICE runtime module that handles requests for services coming from a browser.

CSS
See cascading style sheet.

current result
(1) A temporary Cool ICE report. (2) In Data Wizard, refers to the latest available contents of a temporary report that can be used as a data source for application components during a work session.

D

data service
A type of Cool ICE service created by the Data Wizard. Data services retrieve and manipulate data from external databases.

data set
A copy of records retrieved from one or more data sources. In a database update, any changes made to the data set are written back to the data source. *See also* recordset.

data source
A table of information from an existing report, a script-generated report, an SQL query to a relational database table, an ADO data set, or a current result. *See also* current result.

Data Wizard
A Cool ICE development tool that enables service developers to connect and retrieve data from an external database without having to know Cool ICE Script or HTML. Using the Data Wizard, the service developer creates a data service that can be called directly or from an active service.

department
A group of users specified by a number. By default, users are placed in department 7. Different departments can have different access permissions.

DGate
See Open/OLTP DCOM/OLE Gateway.

DGateAce
A transaction gateway utility that simplifies the process of defining the DCOM client and server programs used with DGate. It creates and saves interface definition files and function definition files that are the basis for the client and server code modules accessed by DGate.

Distributed Component Object Model (DCOM)
Additions to the Component Object Model (COM) that facilitate the transparent distribution of objects over networks and over the Internet. DCOM is part of the specification managed by The Open Group for deployment across heterogeneous platforms.

drawer
In the Cool ICE repository, a group of reports in a cabinet. Each drawer of a cabinet is identified by a letter from B to I; for example, drawer D in cabinet 200. All cabinets in the repository can access drawer A.

dynamic page
An HTML page that the Web server builds each time it is requested by a browser. For example, the page can be built from the latest values in a database. Dynamic pages often contain scripting. *Contrast with* static page.

dynamic service
In Cool ICE, an object that builds a new HTML page every time it is requested by a browser.

E

e-business
The conduct of business transactions over the Internet, such as buying and selling goods, providing customer service, and working with business partners.

e-commerce
The buying and selling of goods over the Internet, especially the World Wide Web.

environment variables
(1) In a workstation environment such as Windows, names that specify global values. (2) In a gateway, a variety of data concerning an Internet HTTP request. These variables are made available to user-written applications when they are requested.

error log
In Cool ICE, the record of all services that failed to execute because of a syntactical error. The Event Viewer menu in ICE Admin allows administrators and service developers to view the error log.

Event Viewer
An option in ICE Admin that lets administrators and service developers view events logged by Cool ICE, such as requests for services made from a browser and service errors.

extranet
The portion of a corporate intranet that is open to customers, business partners, and anyone to whom the corporation grants access privileges.

F

field
An item in a database record, for example, Name, City, Zip Code, and so on. A field may have a specific number of characters or may vary. A group of fields comprise a record. *See also* column, record, and table.

form
(1) A part of a Web page where users can enter information such as name, address, and phone number. The browser uses this information for other Web pages, or passes the information to a database or another application for further processing. (2) In Data Wizard, developers can create a form that lets a service user update a record.

frames
In HTML, the division of the Web browser's display area into separate sections. Each section is called a frame and can contain its own Web page.

FrontPage
See Microsoft FrontPage.

G

gateway
Conversion software that integrates dissimilar entities such as network protocols, software object models, or data storage devices.

Gateway Configuration Utility
A component of Cool ICE that enables an administrator to create, update, or delete Cool ICE connection pools, configurations, and gateways.

graphics server
The server that creates Cool ICE Graphics.

H

hidden columns
In Data Wizard, existing columns that are not available for processing by the application component.

High-performance Transaction Processing Interconnect (HTP/ic)
System software that provides the underlying connection between Windows NT and the Open/OLTP client/server applications. HTP/ic facilitates communication to the HTP/x product and, ultimately, the Open/OLTP product. HTP/ic is based on XATMI and OSI TP standards.

Hypertext Markup Language (HTML)
The primary language used to define documents accessed over the World Wide Web. As a markup language, HTML uses tags to identify how the text is used (for example, first-level heading, paragraph, list), as opposed to how the text should be presented. The Web browser determines how to present the text, for example, the font, type size, indents, and justification. HTML also allows authors to define hypertext links between locations in the same document and between other documents and multimedia objects.

HTML
See Hypertext Markup Language.

HTML editor
An authoring tool for creating HTML documents.

HTP/ic
See High-performance Transaction Processing Interconnect.

HTTP
See Hypertext Transfer Protocol.

hyperlink
Short for hypertext link. A predefined connection from a Web page to another resource on the Web. The link is displayed as either text or an icon (graphic). The destination of the hyperlink can be another page, a multimedia file, a program, or a service. When a user clicks on a hyperlink, the browser passes the request to the appropriate Web server and waits for it to return an HTML page. The destination of the hyperlink is coded as a URL.

Hypertext Transfer Protocol (HTTP)
The TCP/IP format used on the World Wide Web to exchange HTML documents between a browser and Web server. When a user requests an HTML page by either typing a URL or clicking on a hyperlink, the browser builds an HTTP request and sends it to the appropriate Web server.

I

ICE Admin
The Cool ICE administrative component that manages services and user profiles, organizes services and categories, and configures Cool ICE.

IIS
See Microsoft Internet Information Server.

internet
Any collection of interconnected networks. An internet does not necessarily use TCP/IP.

Internet
The global collection of interconnected networks that use the TCP/IP suite of communications protocols. The Internet provides World Wide Web, Usenet news, and other services to users.

Internet Explorer
See Microsoft Internet Explorer.

intranet
A private network that uses Internet technologies. An intranet is not available to the general public, only to those individuals granted access permission.

J

Java
A platform-independent, object-oriented language derived from C++. It is optimized for the Internet.

JavaScript
An interpreted programming or script language that was developed by Netscape Communications Corporation. JavaScript programs can be added to Web pages to create interactive documents.

JScript
The implementation of JavaScript from Microsoft Corporation, which is built into Internet Explorer. JavaScript and JScript are not 100% compatible.

L

LiveWire
See Netscape LiveWire.

M

MAPPER Administration Program
The program that assists administrators with many different repository administration tasks. These tasks include managing files, sites, and memory usage; backing up the repository, initializing the repository, and so on.

MAPPER Relational Interface (MRI)
The Cool ICE interface to data stored in relational tables. MRI consists of seven commands designed to perform specific relational tasks, the Relational Database Interface (RDI), and software programs to manage the communications between the Cool ICE Engine and the relational database.

MAPPER system
A file management system that enables the user to maintain and manipulate a large amount of data in a report-structured database.

marshal
The process of packaging and sending interface parameters across boundaries in computer memory.

mask character
In the Data Wizard, a character that marks a portion of a column as significant for future steps in the application component. The Data Wizard ignores those portions of the column that are not marked as significant. *See also* column, partial column.

method
Member function of an exposed object that performs some action on an object, such as saving it to disk. A method is a logical operation provided by an object. Operations performed on objects are defined as methods of the object. To invoke a method, an object sends a message consisting of the receiving object and the name of the specific method to invoke. The name of the method is sometimes called a *selector*.

Microsoft FrontPage
A product from Microsoft Corporation for creating and managing Web sites.

Microsoft Internet Explorer
A popular Web browser developed by Microsoft Corporation.

Microsoft Internet Information Server (IIS)
Web server software integrated into the Windows NT Server 4.0 operating environment.

Microsoft Transaction Server (MTS)
A feature of the Windows NT Server 4.0 operating environment that provides a component-based transaction processing system for developing, deploying, and managing Internet and intranet server applications. It combines the features of a transaction processing (TP) monitor and an object-request broker (ORB).

MIME
See Multipurpose Internet Mail Extension.

MRIM
See Relational Interface Manager.

Multipurpose Internet Mail Extension (MIME)
A specification for sending and receiving audio, video, graphics, and other types of nontextual data over the Internet. Browsers recognize different MIME types and can launch an associated helper application when they encounter that MIME type. *See also* Secure Multipurpose Internet Mail Extension.

N

native Cool ICE service
An object in the repository that contains any combination of HTML, client-side scripting, or server-side scripting. The server-side scripting language is CoolICE Script. *Contrast with* open Cool ICE service.

Netscape Communicator
A popular Web browser developed by Netscape Communications Corporation.

Netscape Enterprise Server
High-performance Web server software for creating, managing, and distributing information and online applications.

Netscape FastTrack Server
Entry-level Web server software designed to let users with minimal experience create and manage a Web site.

Netscape LiveWire
A suite of management and development tools that enables programmers to create Web content, sites, and applications for the Netscape Enterprise and FastTrack servers.

Netscape Navigator
A popular Web browser developed by Netscape Communications Corporation.

O

object
(1) In Cool ICE, any item that can be stored in the repository. An object can contain binary or textual data. GIF or JPEG images, applets, Microsoft Word documents, Cool ICE services, and static HTML documents are examples of objects. (2) In object-oriented programming, a self-contained entity that consists of both data and the procedures to manipulate the data.

Online Transaction Processing (OLTP)
A high-performance type of application characterized by a service request from a user (client) immediately followed by an action and response from a server system. A complete request-response cycle is called a transaction. OLTP systems must meet certain criteria, called ACID properties (atomicity, consistency, isolation, and durability), to maintain the integrity of the data and ensure that transactions are executed correctly.

open Cool ICE service
An object in the repository that contains any combination of HTML, client-side scripting, and server-side scripting. The server-side scripting language is Visual Basic Script or JScript. *Contrast with* native Cool ICE service.

Open Database Connectivity (ODBC)
A vendor-neutral interface, based on the SQL Access Group specifications, announced by Microsoft in December 1991. A developer can use ODBC to access data in a heterogeneous environment of relational and nonrelational databases.

Open/OLTP DCOM/OLE Gateway (DGate)
A Unisys software component that enables Distributed Component Object Module (DCOM) clients to access OS 2200 applications.

P

parameter
A value passed to a function, procedure, subroutine, command or program by the caller. *Contrast with* variable.

partial column
In the Data Wizard, a step that lets developers define a significant portion of the column. The Data Wizard only processes the significant characters; the other characters are ignored. When developers define partial fields, uppercase or lowercase x characters mark the significant portion of the column. *See also* mask character.

profile
See security profile.

protocol
A mutually determined set of formats and procedures for the exchange of information between computers.

Q

query definition
A service created by the Data Wizard that specifies the data sources (databases, tables, and columns) you want to access. It also specifies how the data appears and the operations you want performed.

R

record
In a database, a group of columns or fields that make up one complete entry. For example, a record about a user might contain fields for name, address, telephone number, and so on. A group of records makes up a table.

recordset
A temporary set of records that Cool ICE retrieves from a data set using the RetrieveDataSet() method. *See also* data set.

relational database
A type of database management system that stores data in the form of related tables. With relational tables, the same database can be viewed in many different ways. *See also* table.

Relational Interface Manager (MRIM)
A set of programs used to manage the communications between the Cool ICE Engine and the relational database management system. MRIM consists of the Request Manager, which submits the code generated by MRI commands to the database manager and returns data to Cool ICE, and a Database Handler for each database supported by MRI. In addition, a Remote Communications Handler is employed to access data stored on a remote host.

report
The set of data that you work with. Reports are identified by a unique report number.

repository
See Cool ICE repository.

result
A temporary report created by Cool ICE Script commands.

row
A horizontal set of data, as in a relational database table. *See also* record, column, and table.

S

script
A series of instructions or commands that are interpreted rather than compiled.

scripting language
A programming language for writing scripts. VBScript, JavaScript, and Cool ICE Script are examples of scripting languages.

Secure Electronic Transactions (SET)
A protocol for securing credit card transactions on the Internet. SET uses digital signatures to verify the identity of customers.

Secure Multipurpose Internet Mail Extension (S/MIME)
A standard for sending and receiving encrypted mail.

Secure Sockets Layer (SSL)
A protocol that allows for the secure transmission of confidential information between a client and a server. SSL uses a public key to encrypt data being sent over the Internet. Both Netscape Navigator and Microsoft Internet Explorer support SSL.

security profile
A name associated with a list of user-ids and then allocated to one or more categories, services, or objects in the Cool ICE Repository. When a site needs to restrict access to a particular application, the administrator uses ICE Admin to create a security profile and assign the user-ids of those users who are permitted to access that application. (Only user-ids that have been registered in ICE Admin can be assigned to a security profile.)

When the security profile is defined, the administrator allocates or assigns it to the application. Only users whose user-ids are in the security profile can access that application.

server side
Refers to any operation that is performed at the server in a client/server relationship. *Contrast with* client side. For example, Cool ICE Script executes on the server.

service
See Cool ICE Service.

services table
In Cool ICE, a list of all objects and dynamic services assigned to a category. Each category has its own services table.

session
A lasting connection between the browser and the Web server involving the exchange of data. A Cool ICE configuration parameter determines the length of the session. *Contrast with* connection.

site
In Cool ICE, the combination of the repository, its list of users, and its general operations. One computer system can contain up to 26 sites.

SQL
See Structured Query Language.

standard application component
An application component that queries a database, manipulates the data, and possibly defines a data format. A standard application component never includes steps that change a data source. *See also* application component, step. *Contrast with* transaction application component.

state
For an HTML page, the set of current or last-known values for its attributes, configuration, or content. "Maintaining state" refers to keeping track of this data over time. To maintain state information over time, some Web sites send cookies to the user's workstation.

state management
A feature that allows a Cool ICE application to preserve and maintain information on the server between multiple dialogs with the browser user. The State Management feature provides functions to save, restore, and remove information created and maintained by an application. It also performs a scheduled automatic cleanup of information left over from sessions that have timed out.

state management report
The Cool ICE report that includes state information. *See also* state management.

stateless
Not maintaining state information over time. All Internet requests are stateless by default.

static page
An HTML page that the Web server returns each time it is called by a Web browser. The Web server does not change or modify the content of the page. *Contrast with* dynamic page.

static service
In Cool ICE, an object that contains HTML, but not any scripts. A Web server returns the same HTML page each time it executes a static service.

step
In the Data Wizard Component Builder, the smallest editable unit of work that can be performed in a component. Examples of steps include Select source, Perform calculation, and Sort.

step data
One or more temporary reports that Data Wizard keeps in state management for the purpose of building an application component.

Structured Query Language (SQL)
An industry-standard set of relational commands that let a user create, retrieve, and update data in a relational database.

style guide
In Cool ICE, the variables used to maintain a consistent layout and appearance of the HTML pages created by a service. It also provides a method for maintaining common variables used in dynamically generated HTML pages.

style sheet
See cascading style sheet.

T

table
A presentation of information organized in rows and columns or fields. A database can show intricate relationships between and among fields and databases.

trace log
A debugging aid in Cool ICE that contains service requests and input captured from the browser. The Event Viewer menu in ICE Admin allows administrators and service developers to view, run, and delete trace logs.

transaction application component
An application component that changes a single record in a single database. It includes a setup part that accepts the update values and an action part that performs the record update. *See also* application component. *Contrast with* standard application component.

Tuxedo
A software product based on the X/Open model for distributed transaction processing (DTP). Tuxedo and other X/Open compliant DTP software allow organizations to develop applications that execute transactions across multiple systems and database types.

TWIP
A unit of typographical measurement that is equal to 1/20th of a printer's point. There are 1,440 twips to an inch, or about 567 twips to a centimeter. TWIP is the unit of measurement for Cool ICE Graphics.

U

Uniform Resource Locator (URL)
A standard for specifying the exact location of a resource, such as a file, on the Internet. URLs are used in HTML documents to specify the target of a hyperlink. The URL includes the name of the protocol needed to access the resource, a domain name, and, for a file, the path to its location on the Web server.

URL
See Uniform Resource Locator.

V

variable
(1) A symbol or name that stands for a value. (2) Independent data item that developers can use in a script or Data Wizard screen to pass single input and output data items to or from a script. Variables can also pass information between statements in a script. *Contrast with* parameter.

Visual Basic Scripting Edition (VBScript)
An interpreted scripting language that is a product of Microsoft Corporation. VBScript is a subset of the Visual Basic Programming Language and is implemented in Microsoft Internet Explorer and Internet Information Server (IIS).

W

Web application
A software program that uses HTTP for its core communications protocol and delivers Web-based information to the user in the HTML language. Also called a Web-based application.

Web browser
A client program used to locate and display HTML pages. The Web browser uses Hypertext Transfer Protocol (HTTP) to request HTML pages from a Web server. Netscape Navigator and Microsoft Internet Explorer are examples of Web browsers.

Web page
A file accessible on the World Wide Web that is embedded with HTML tags, scripts, images, and other technology supported by Web browsers.

Web server
The hardware, operating system, Web server software, communications protocols, and Web site content that is connected to the Internet.

Web server software
The software running on the Web server that returns HTML pages in response to an HTTP request from a browser. Netscape Enterprise Server, Netscape FastTrack Server, and Microsoft Internet Information Server are examples of Web server software.

Web site
A location on the World Wide Web that is usually a related collection of Web pages. Each Web site contains a home page from which a user can access all the other pages on that site.

Web variables
(1) A source of input from the browser such as values from a form. (2) In Data Wizard, specific variables identified during the Create variables step. *See also* variable, form.

WHERE clause
In Data Wizard, the set of conditions that a data source record must meet in order to be selected for use in an application component. *See also* data source, record.

WorldPay
A secure, multicurrency payment system, developed by WorldPay Limited, for use on the Internet.

World Wide Web (WWW)
A global, hypertext-based information system providing an easy way to access documents. It gives users on computer networks a consistent way to easily access a variety of media.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X